

# PM006 – 21 November 2023

## Update on PINNs

Application to Urban Wind Field Dispersion Studies

# Scripts v3 – Preliminary Results

# Some Updates to v3

- 1) Number of neurons customizable from configuration file (PM003)
- 2) Type of activation function customizable (PM003)
- 3) Type of scaler customizable (PM003)
- 4) Custom points for physics loss (PM004)
- 5) Use of both optimizers (Adam first for an ansatz then LBFGS to fine tune solution) (PM004)
- 6) Option to overlay plots with quiver arrows in configuration file (PM005)
- 7) Option to plot for any angle in configuration file (PM005)
- 8) Option to plot Turbulent Viscosity (with directional arrows) (PM005)
- 9) Loss vs Epochs Plot (PM006)
- 10) RANS added to PINN (PM006)
- 11) Angles to add / remove from training dataset customizable (PM006)

# Some Parameters

Infinite epochs - instead the criteria for stopping is  $\text{loss}_{\{n\}} - \text{loss}_{\{n-1\}} < \epsilon$  for 10 consecutive epochs where  $n$  is the epoch number and  $\epsilon = 1E-5$  (user defined)

128 Neurons for the PINN unless otherwise specified

We have the data for 8 angles, [0, 30, 60, 90, 120, 135, 150, 180] in degrees

We concatenate the data for angles = [0, 30, 60, 90, 120, 150, 180] and then take 99.99% of the dataset with random seed = 42 for training and 0.01% for testing

By using the whole dataset we hope to make the NN learn about wind angle such that the parameters become functions of the wind angle

Then using the trained neural network we predict the data for angle = 135

Progress so far - Data Loss Only  
Standard Normal Scalar  
(Adam Optimizer)

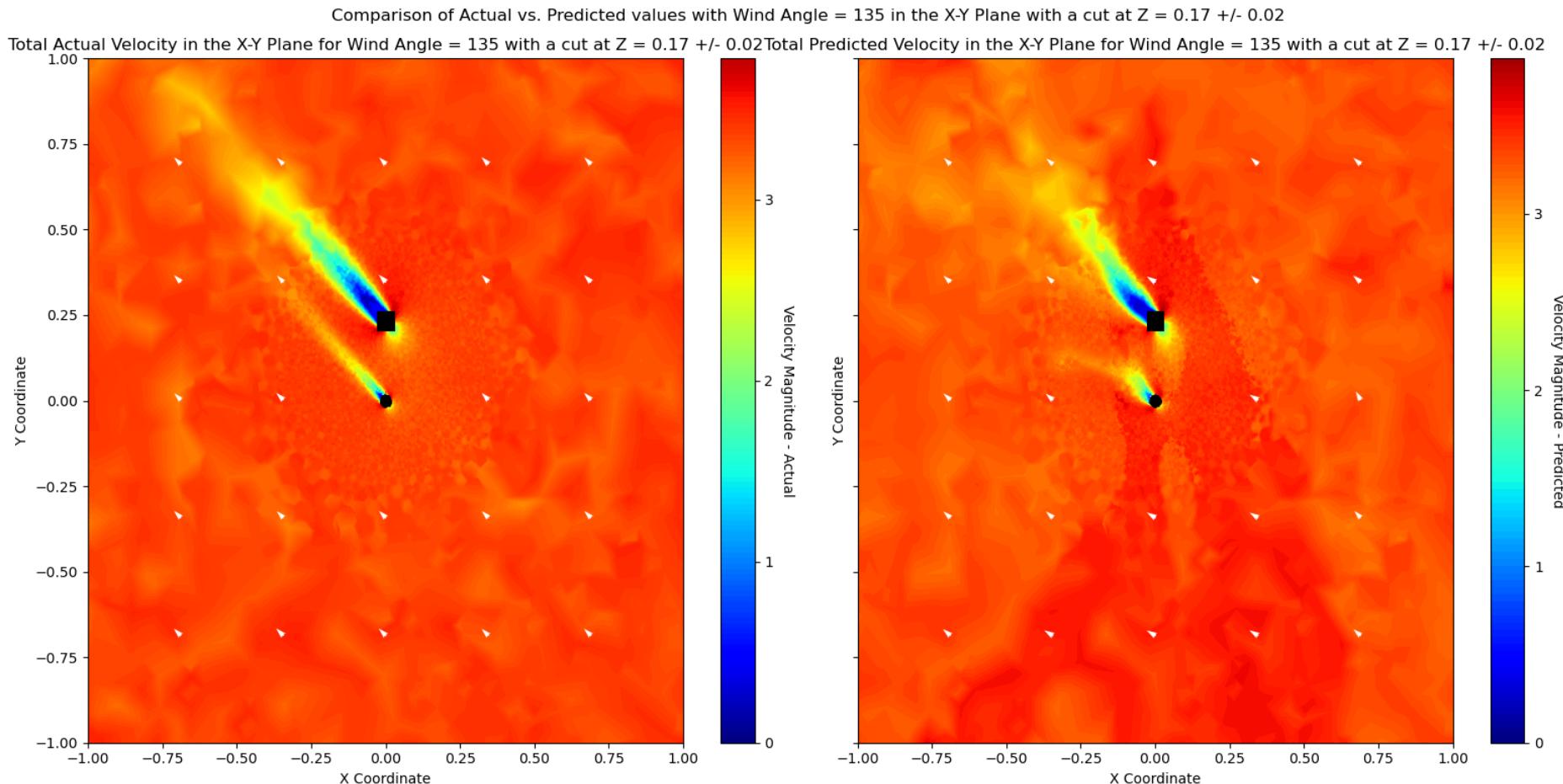
Threshold = 1E-5 (41790 Epochs, not completed), GPU Laptop

Scripts v3 – PREDICTING (135 DEG)

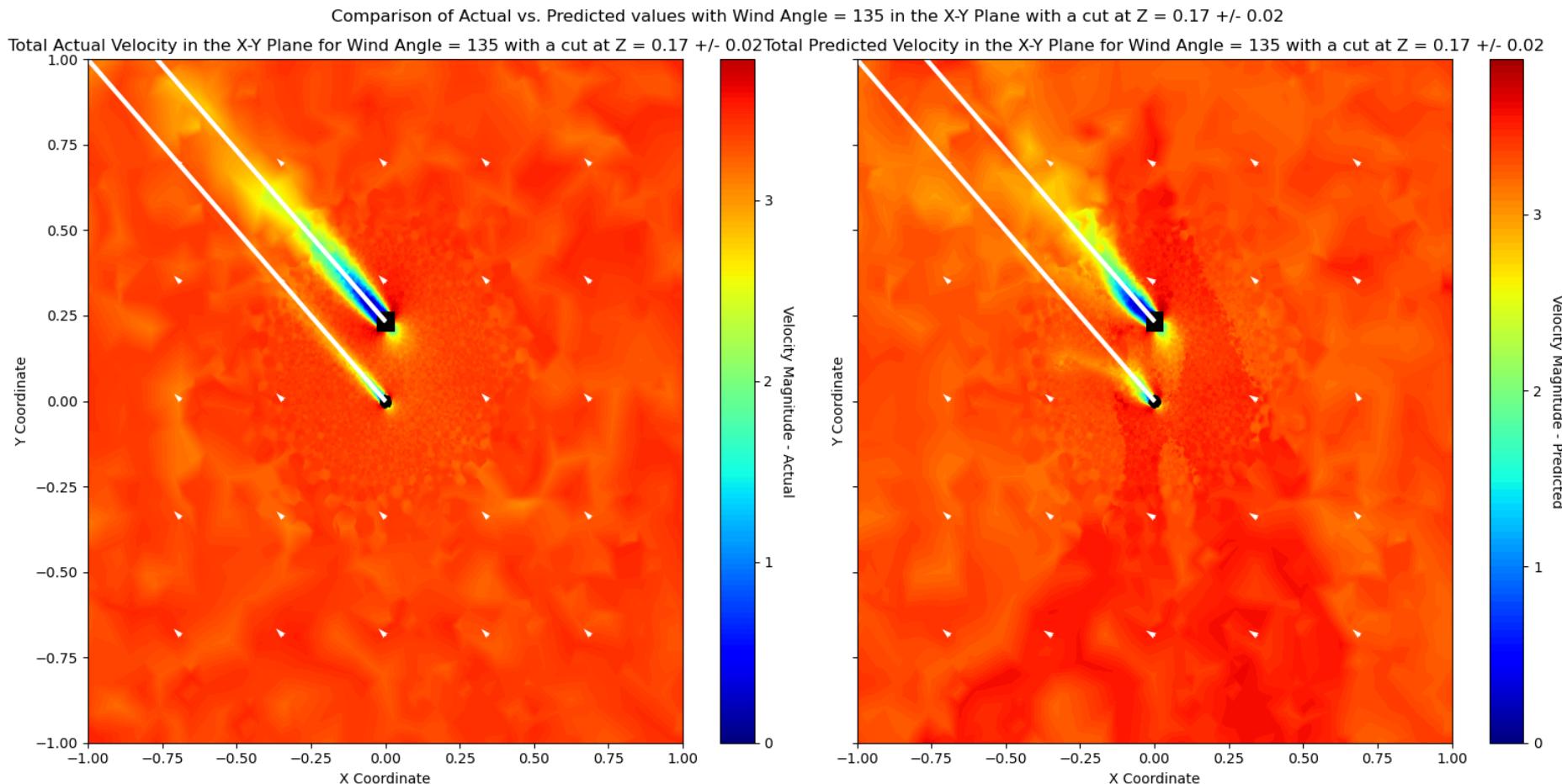
Progress so far - Data Loss Only (Adam Optimizer – Std Normal Scalar)  
Threshold = 1E-5 (41790 Epochs, so far...), GPU Laptop  
Predicting Results – Metrics (Angle = 135)

Variable	MSE	RMSE	MAE	R2
Pressure	0.989688484	0.994830882	0.943660541	0.414511051
Velocity:0	0.321211131	0.566754912	0.488169656	0.684848359
Velocity:1	0.262792097	0.512632516	0.443566786	0.744623474
Velocity:2	0.002707722	0.05203578	0.021826528	0.917230623
TurbVisc	0.235428891	0.485210151	0.352722459	0.998284438

Progress so far - Data Loss Only (Adam Optimizer – Std Normal Scalar), Threshold = 1E-5 (41790 Epochs, so far...), GPU Laptop  
Predicting Results - X-Y Total Velocity Plot (Angle = 135)

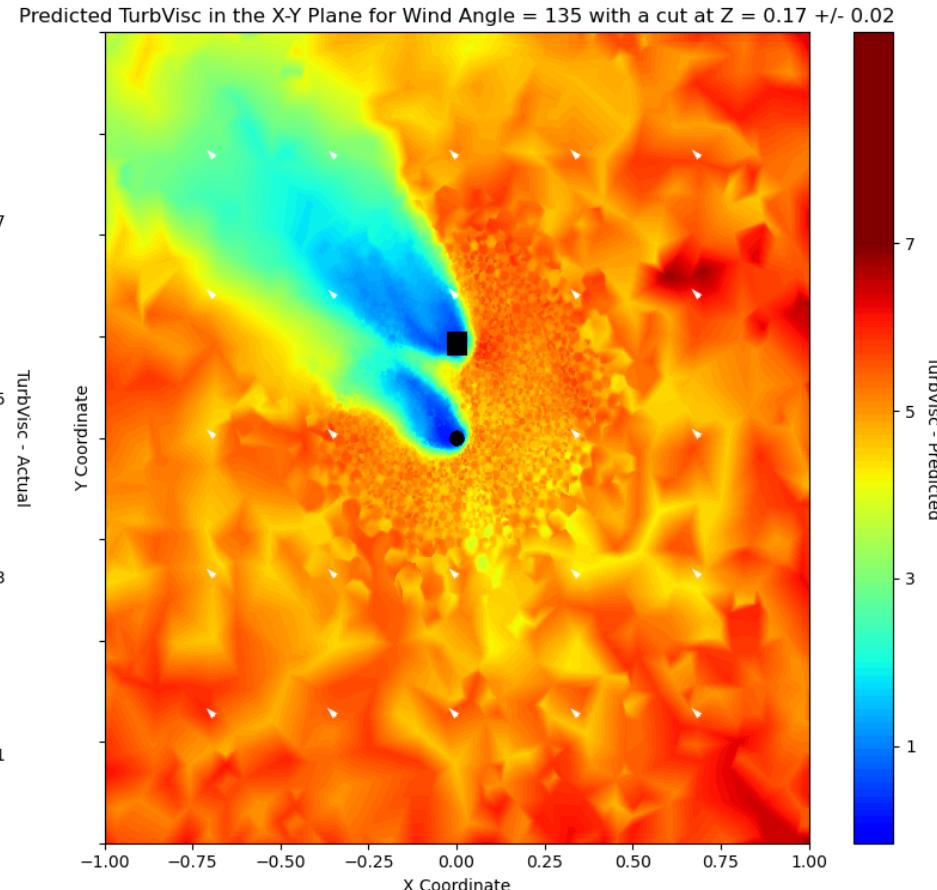
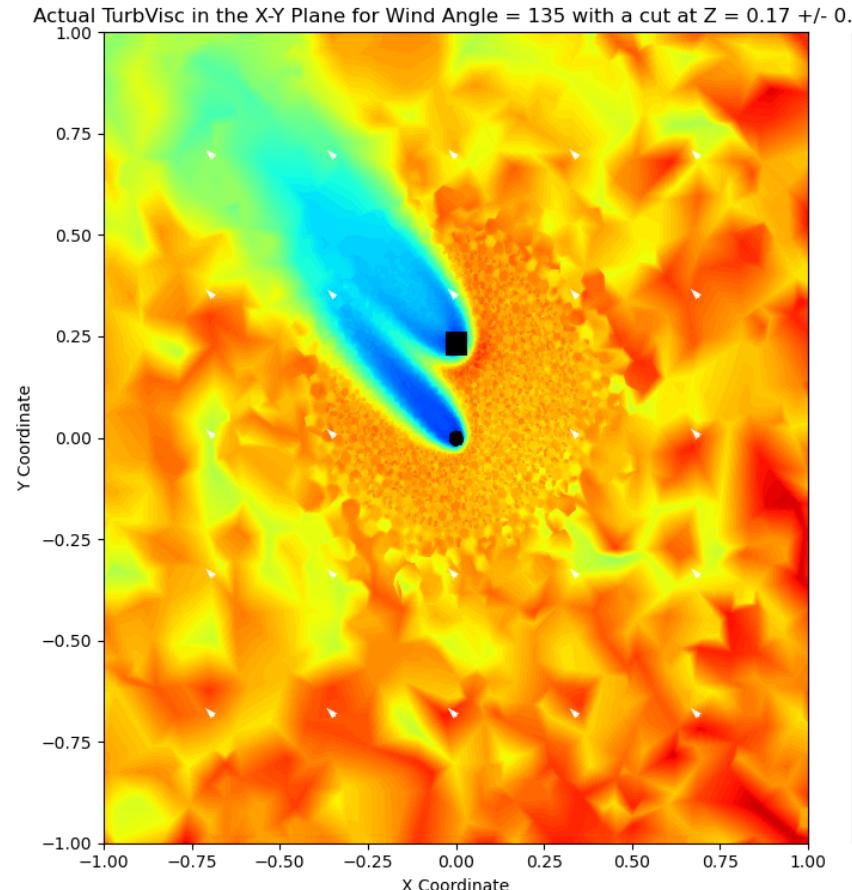


Progress so far - Data Loss Only (Adam Optimizer – Std Normal Scalar), Threshold = 1E-5 (41790 Epochs, so far...), GPU Laptop  
Predicting Results - X-Y Total Velocity Plot (Angle = 135)



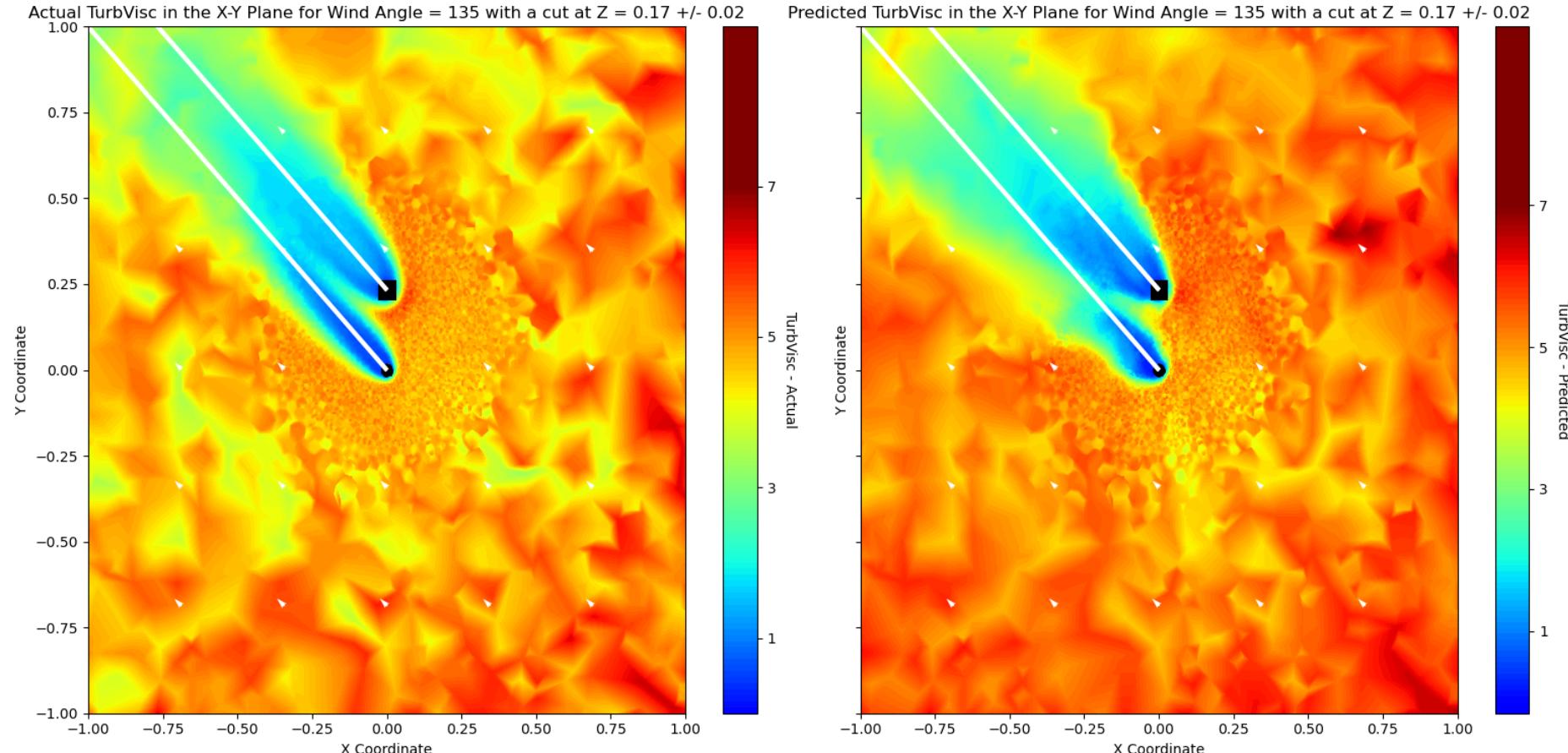
Progress so far - Data Loss Only (Adam Optimizer – Std Normal Scalar), Threshold = 1E-5 (41790 Epochs, so far...), GPU Laptop  
Predicting Results - X-Y TurbVisc Plot (Angle = 135)

Comparison of Actual vs. Predicted values with Wind Angle = 135 in the X-Y Plane with a cut at Z = 0.17 +/- 0.02



Progress so far - Data Loss Only (Adam Optimizer – Std Normal Scalar), Threshold = 1E-5 (41790 Epochs, so far...), GPU Laptop  
Predicting Results - X-Y TurbVisc Plot (Angle = 135)

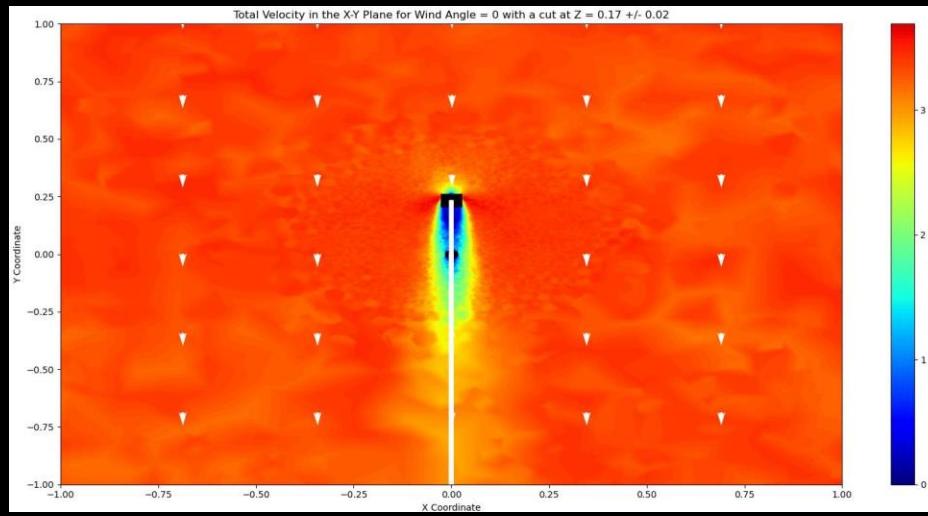
Comparison of Actual vs. Predicted values with Wind Angle = 135 in the X-Y Plane with a cut at Z = 0.17 +/- 0.02

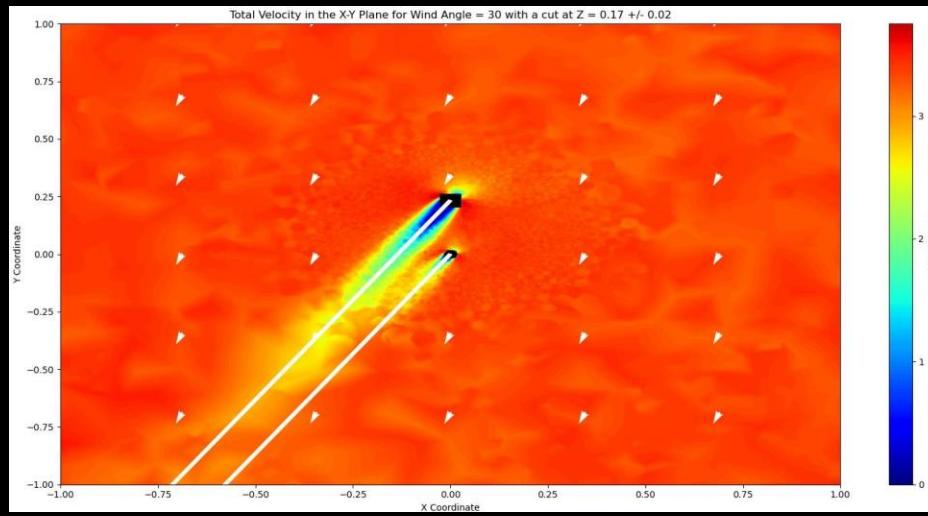


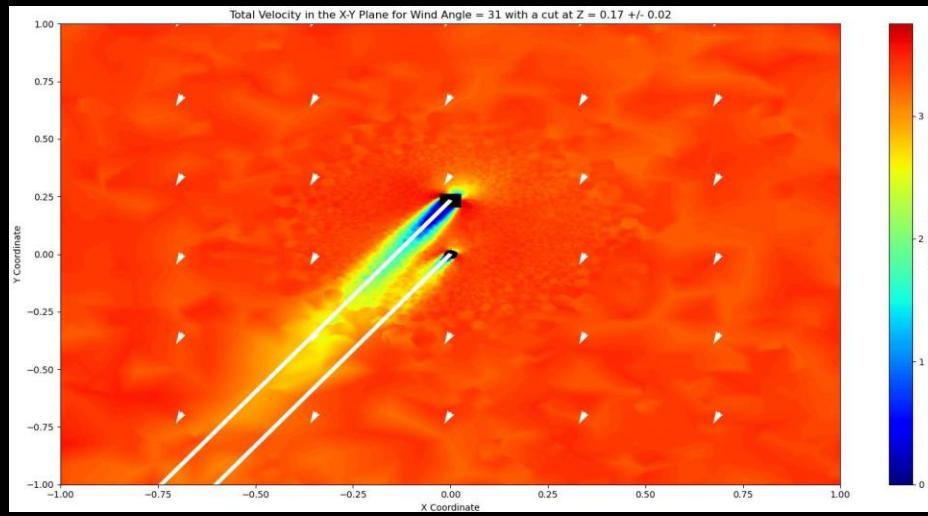
Progress so far - Data Loss Only  
Standard Normal Scalar  
(Adam Optimizer)

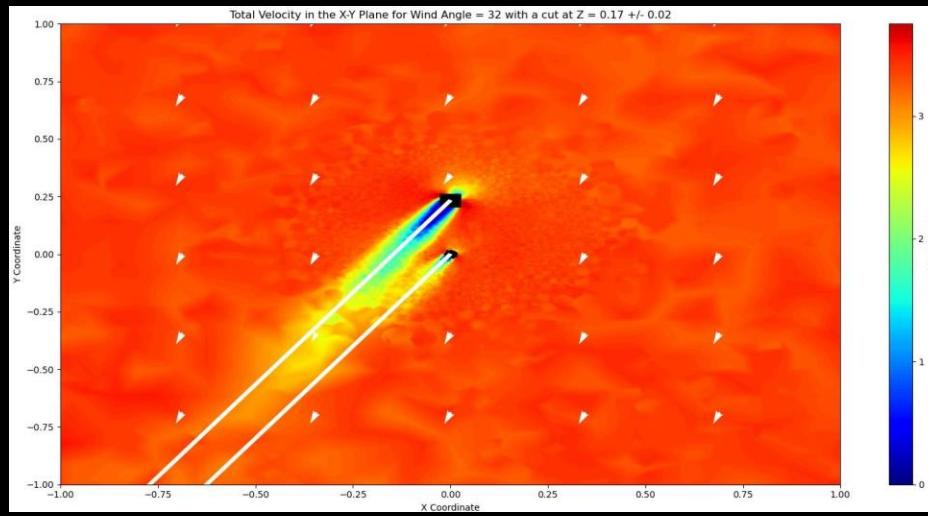
Threshold = 1E-5 (41790 Epochs, not completed), GPU Laptop

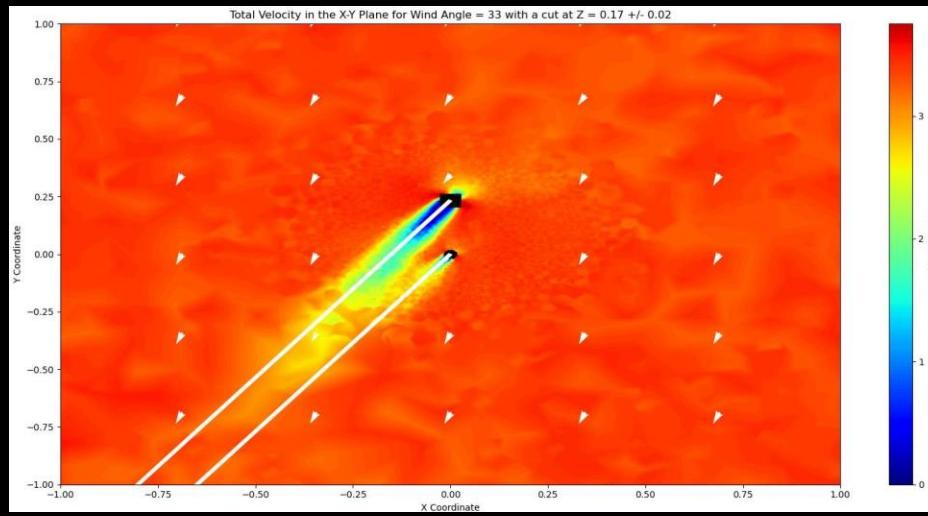
Scripts v3 – Plotting Any Angle

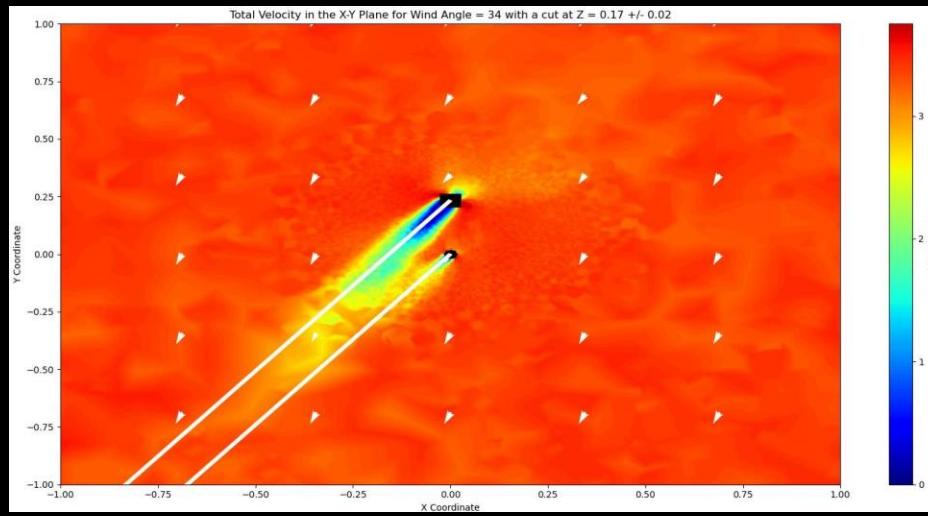


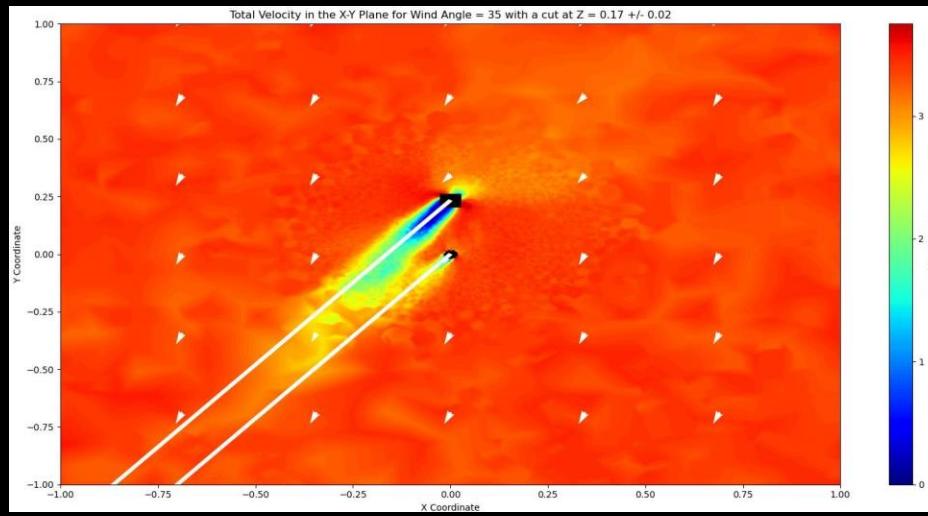


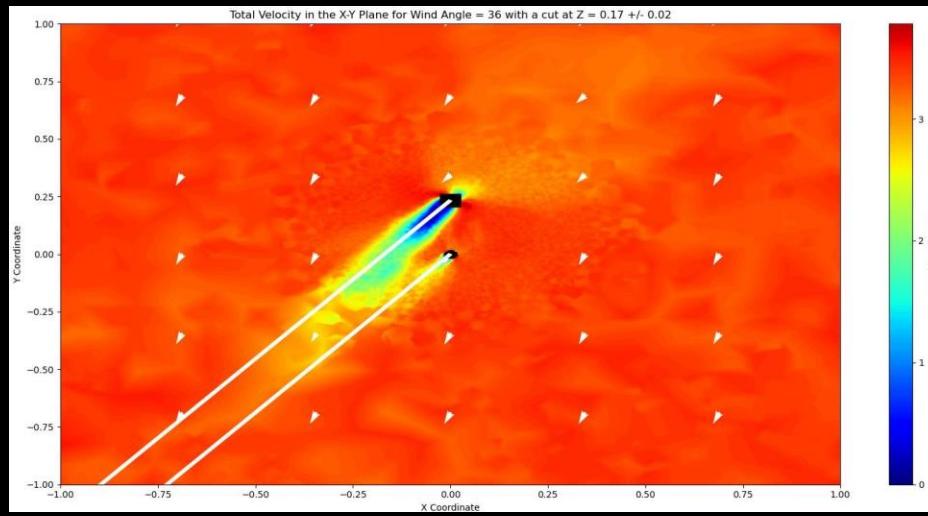


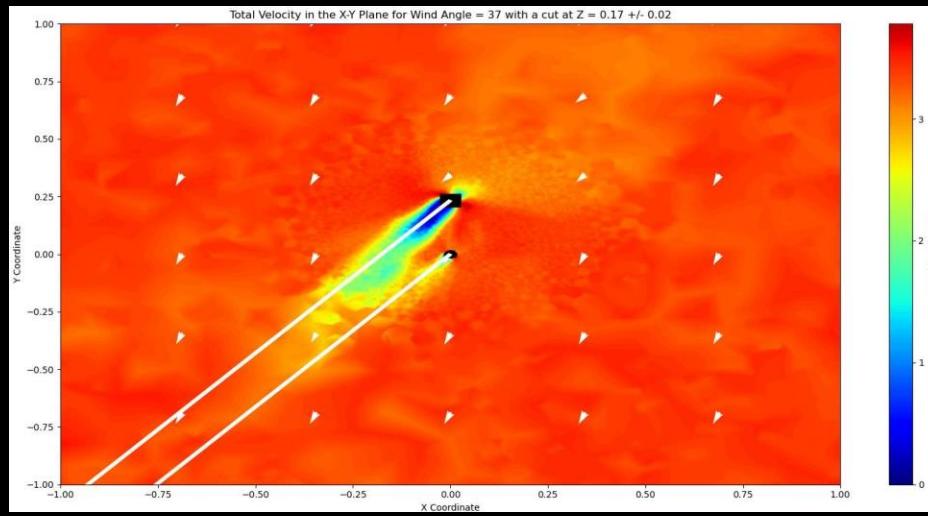


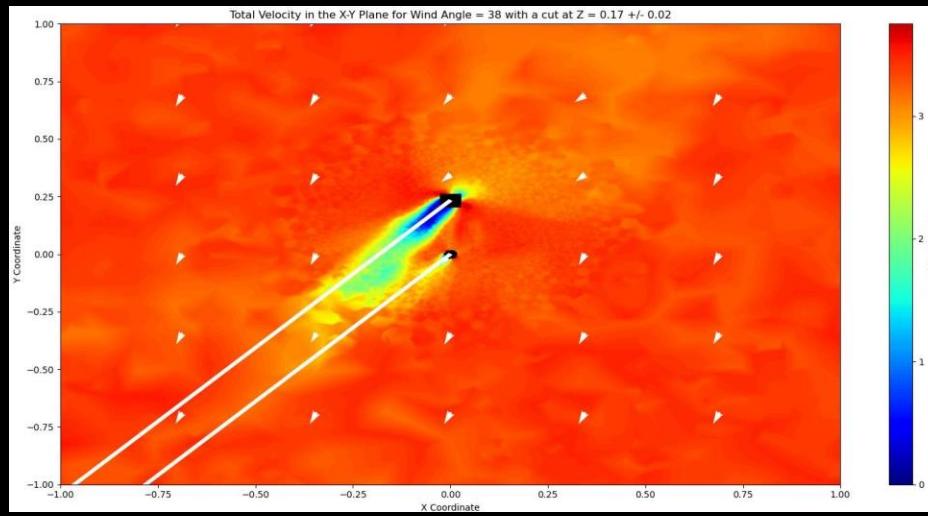


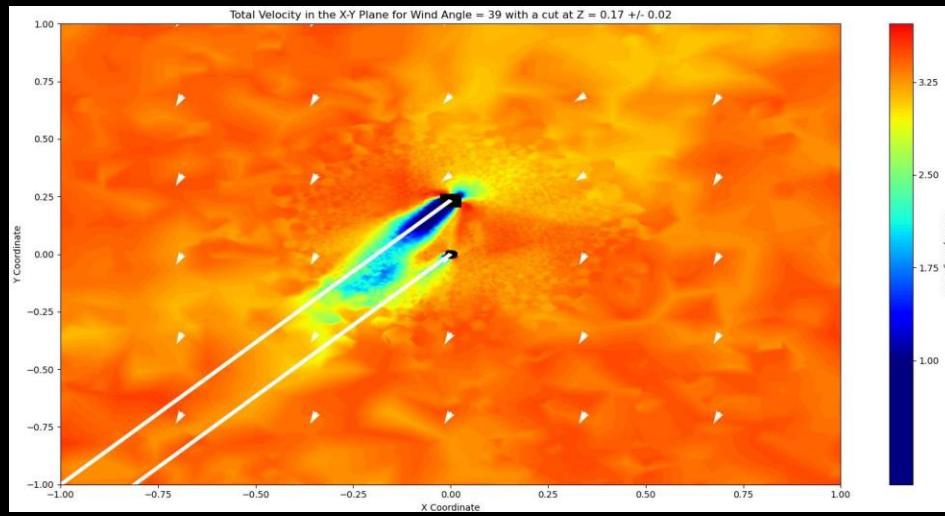


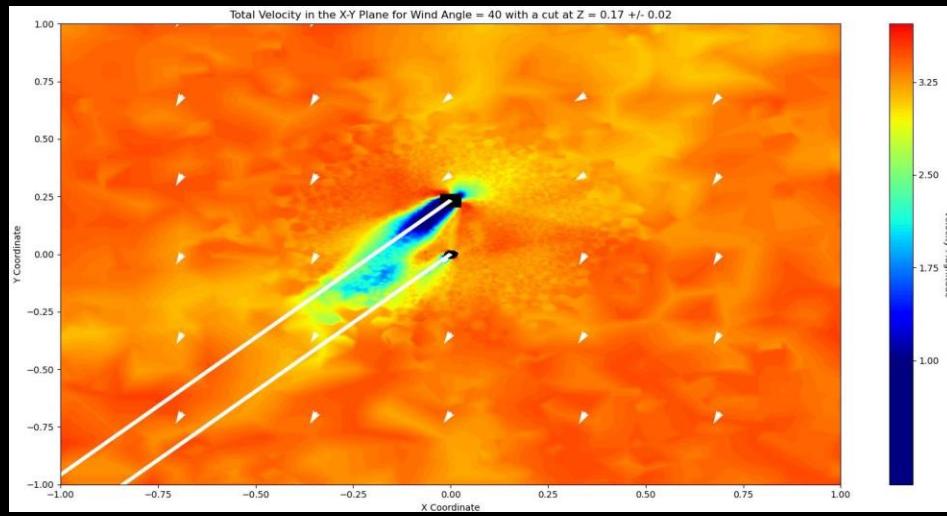


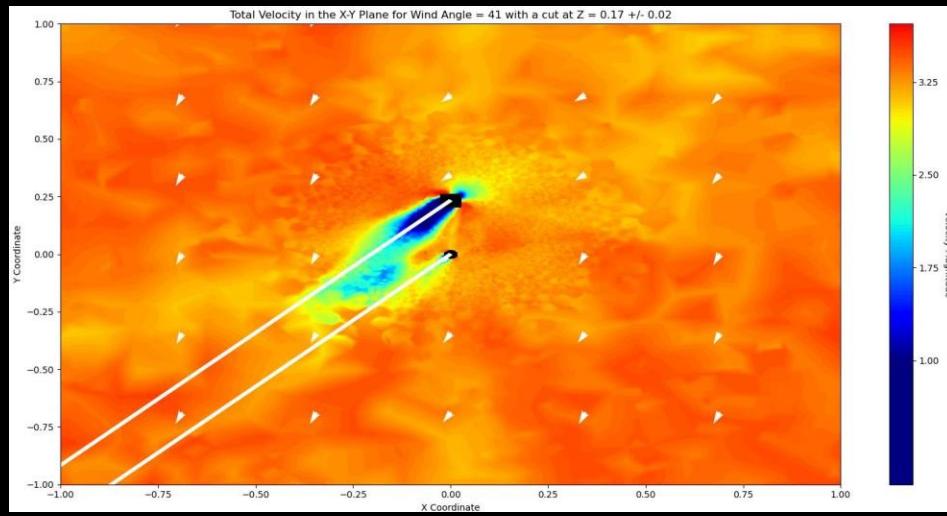


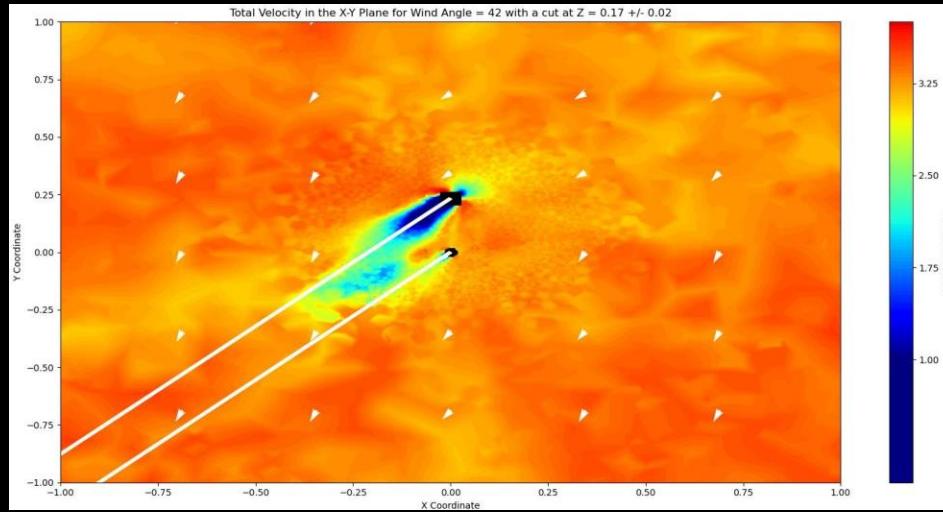


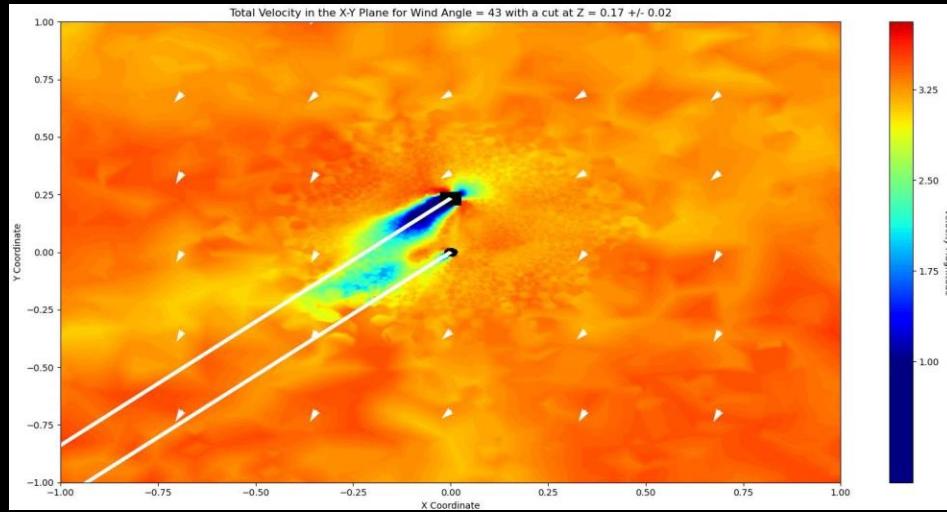


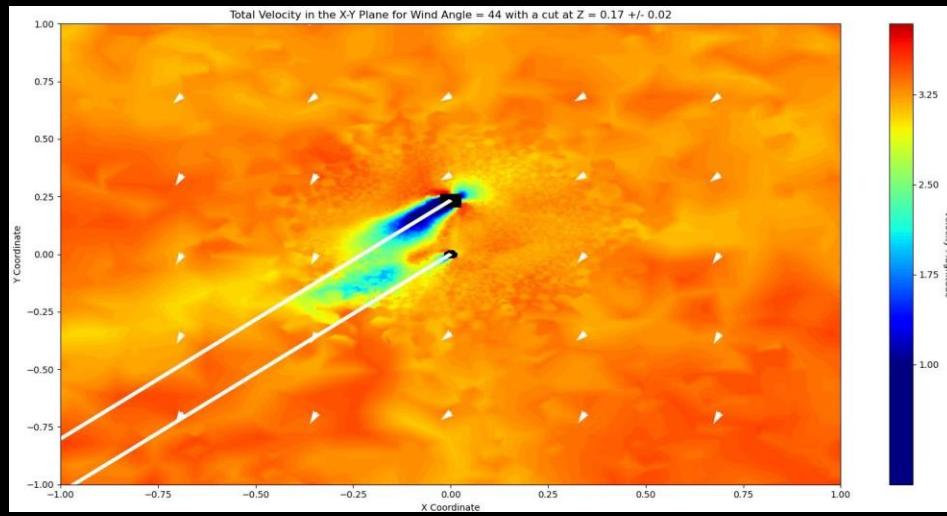


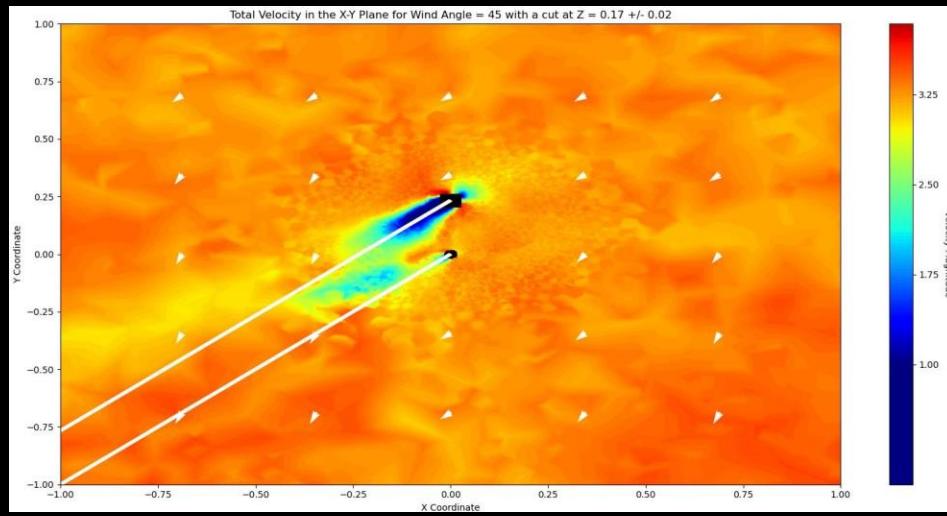


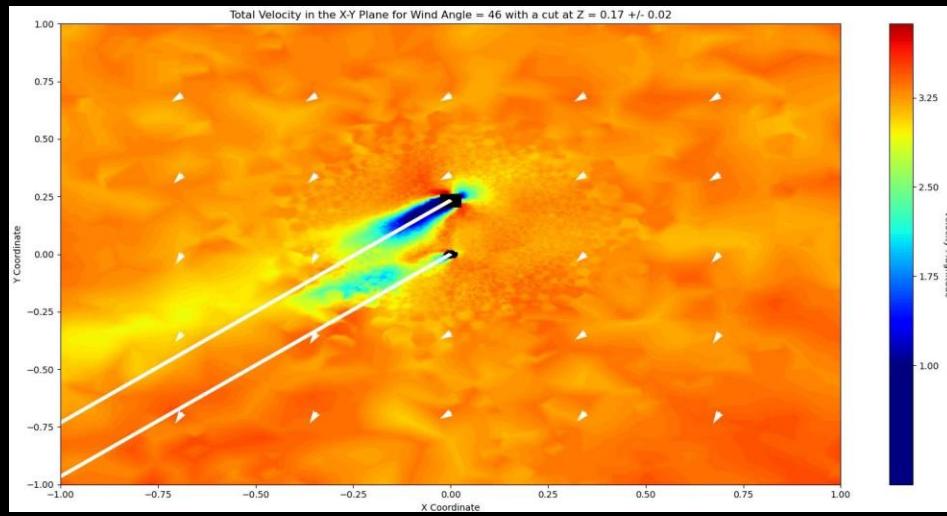


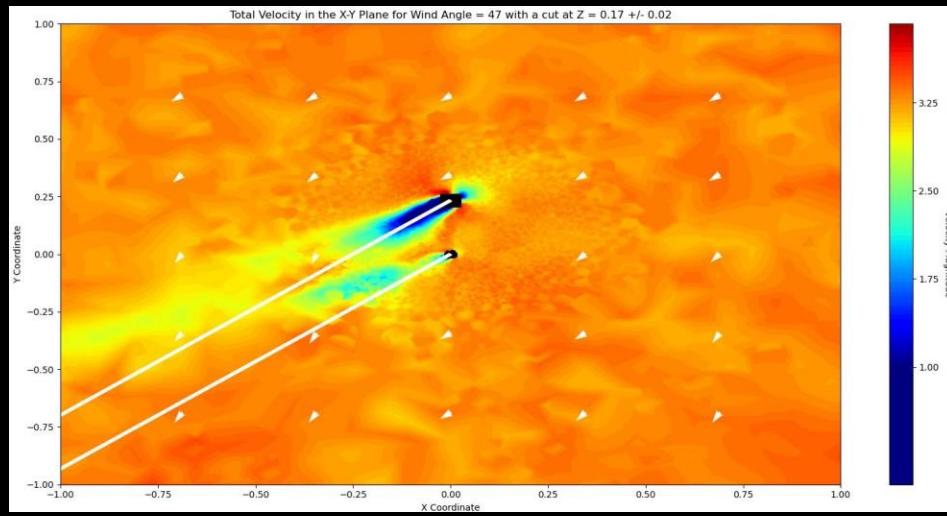


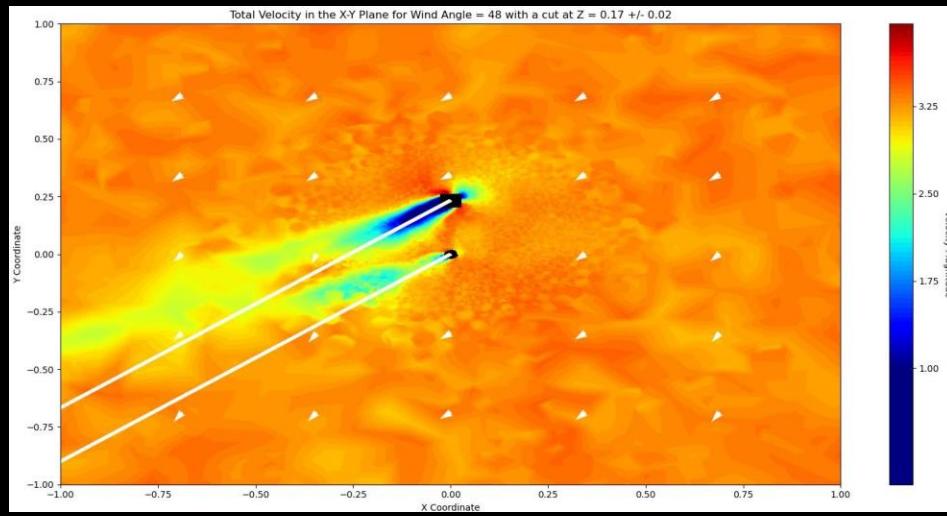


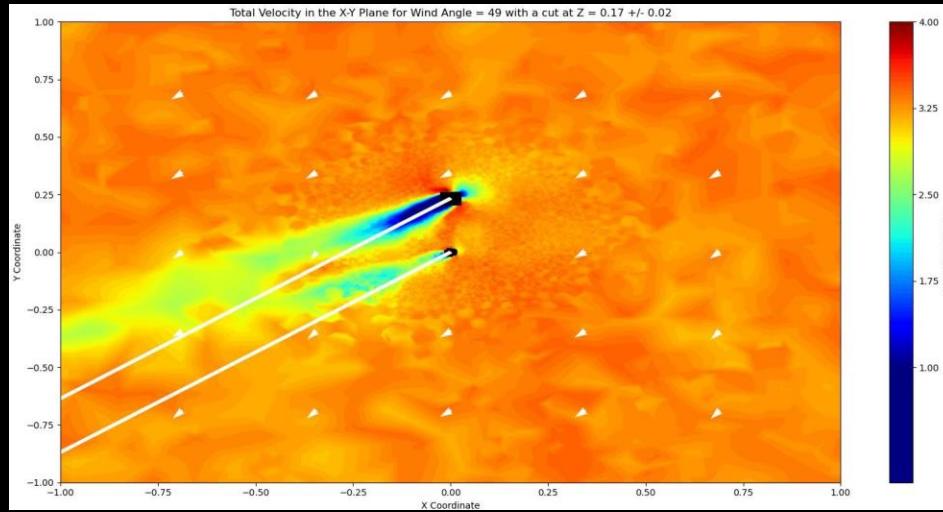


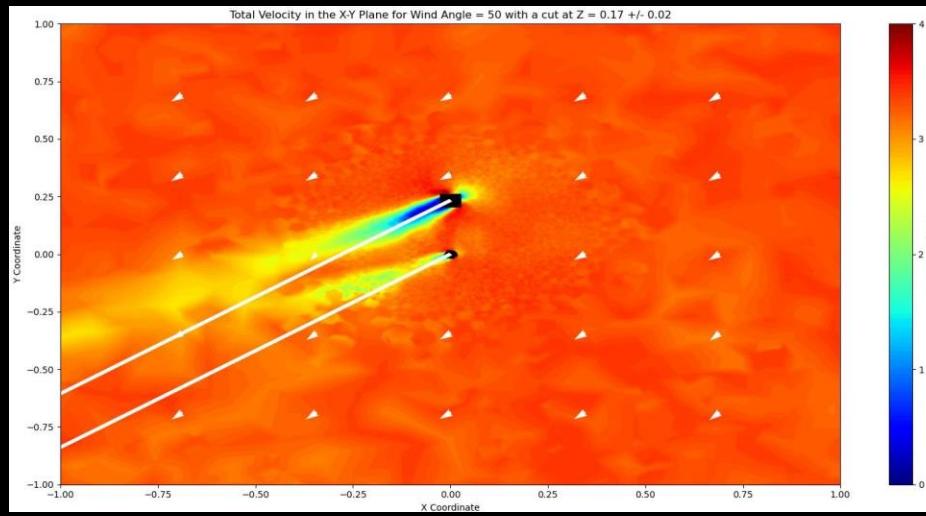


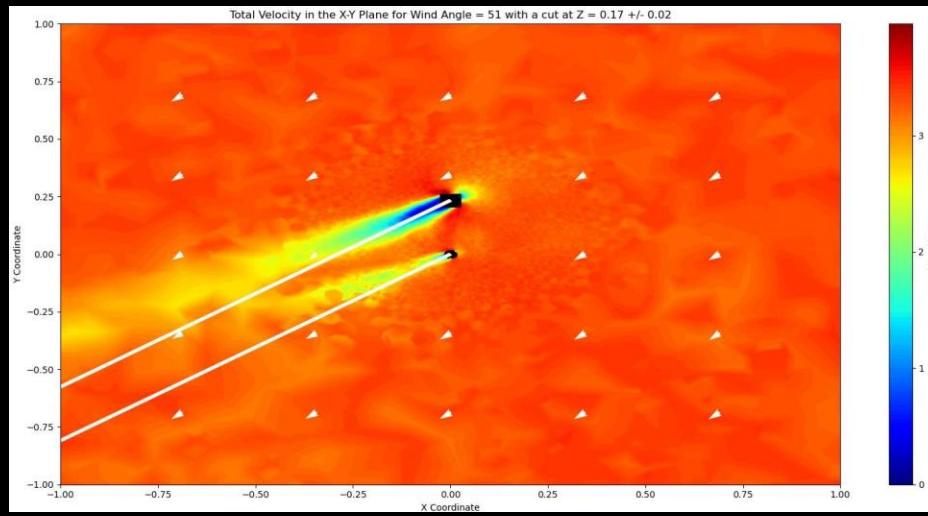


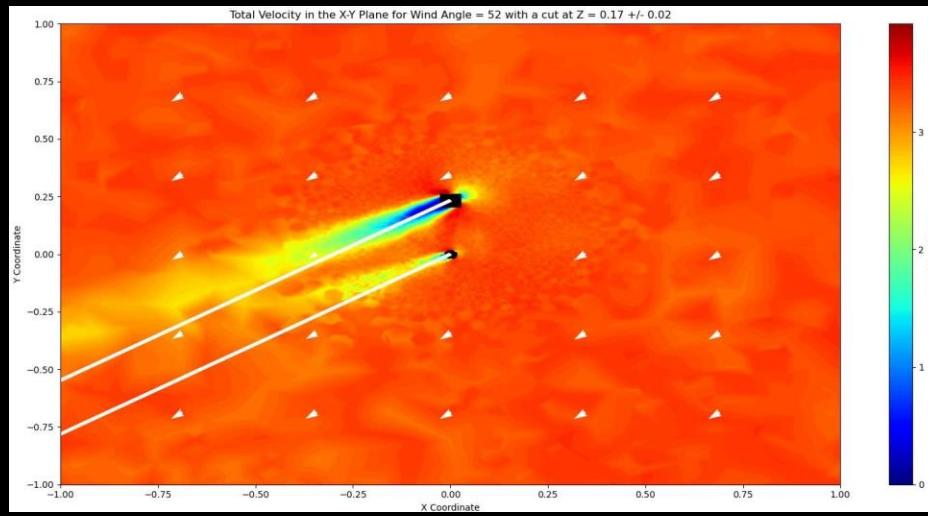


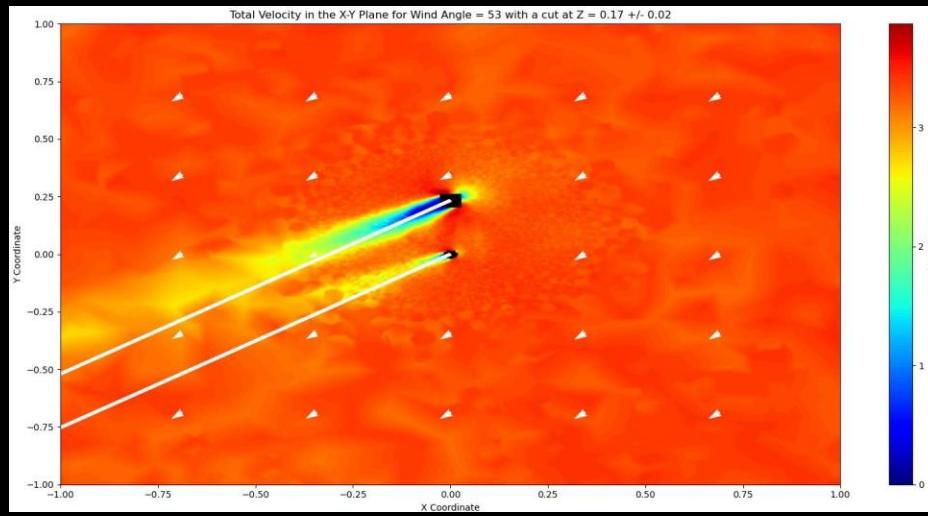


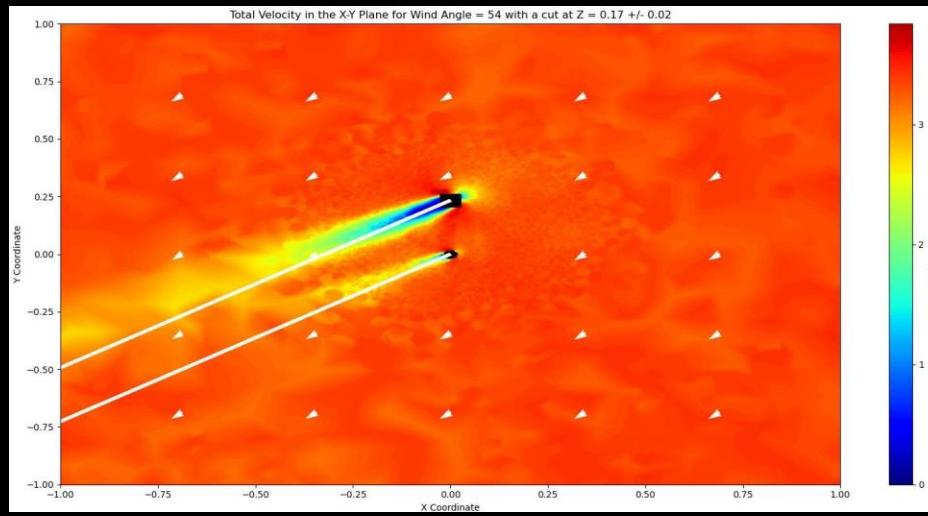


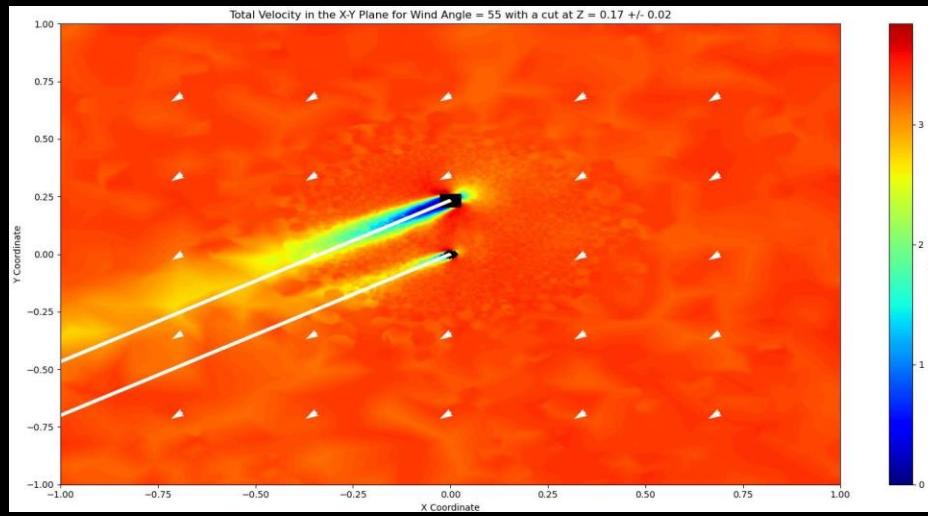


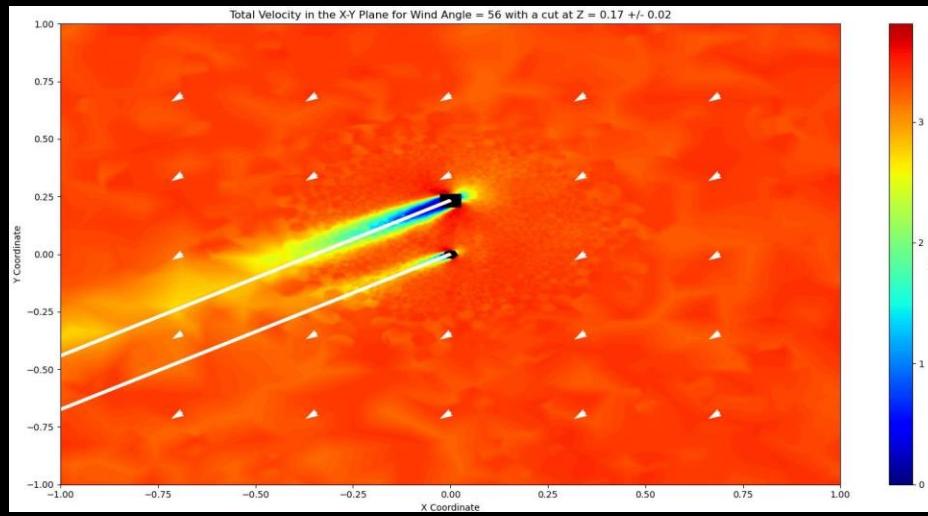


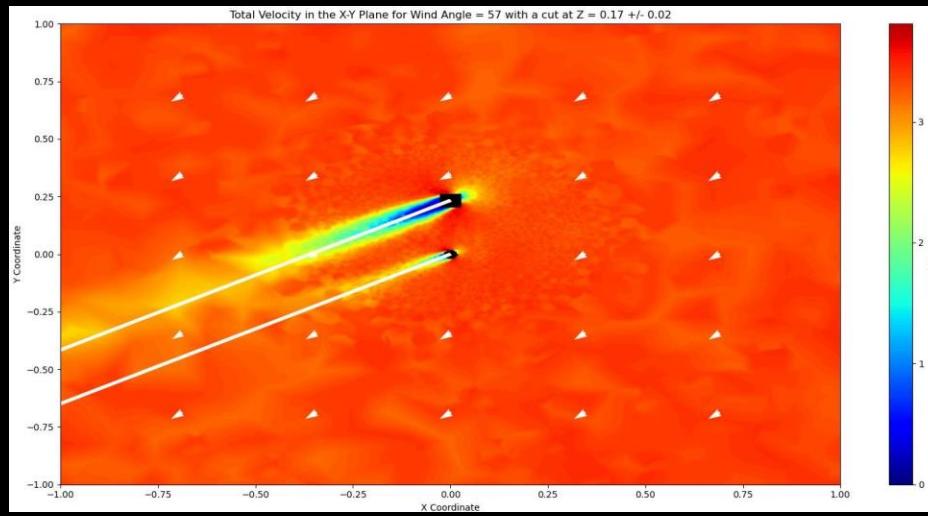


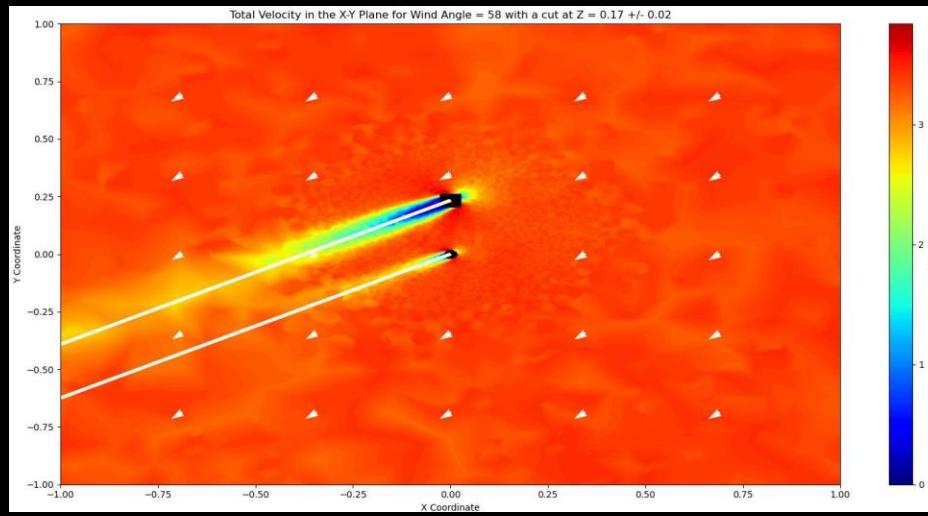


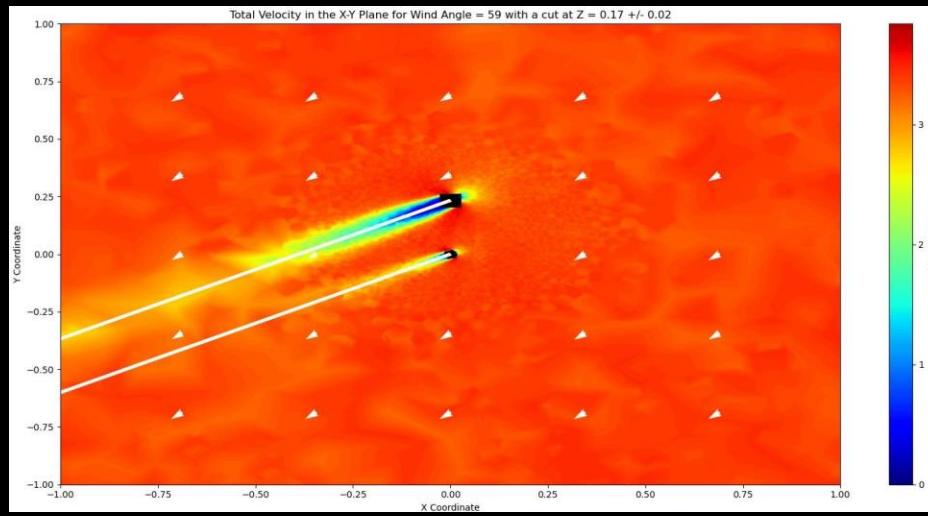


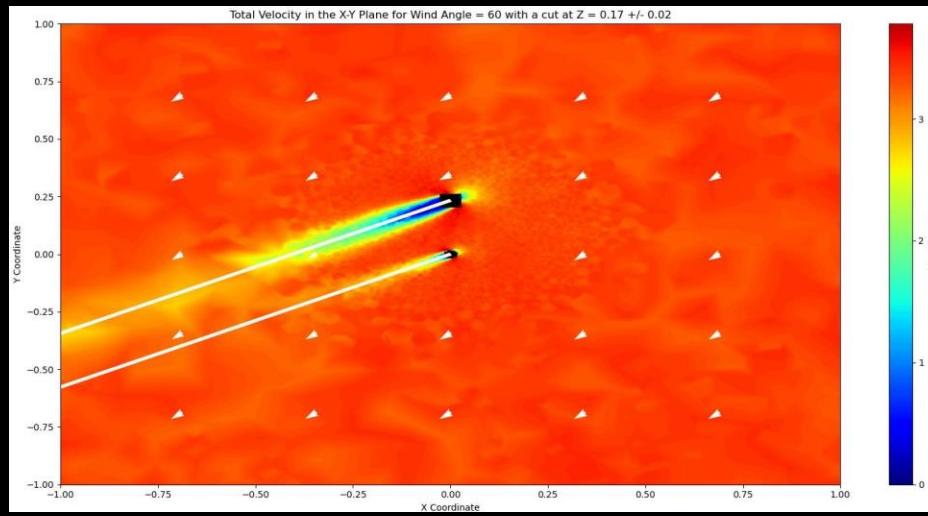


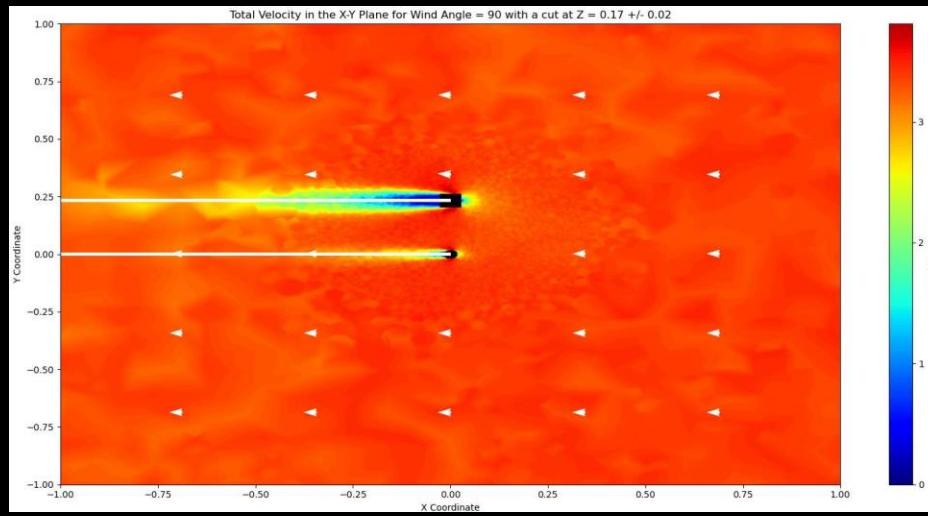


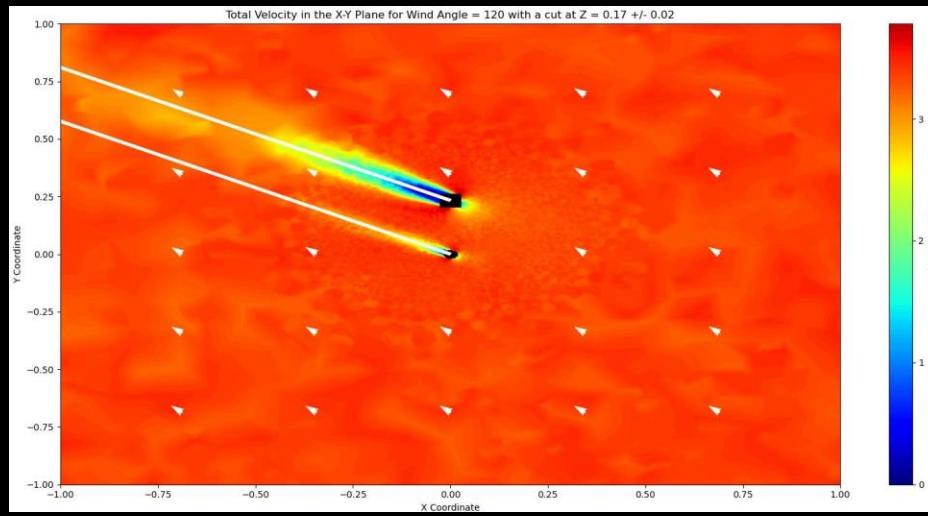


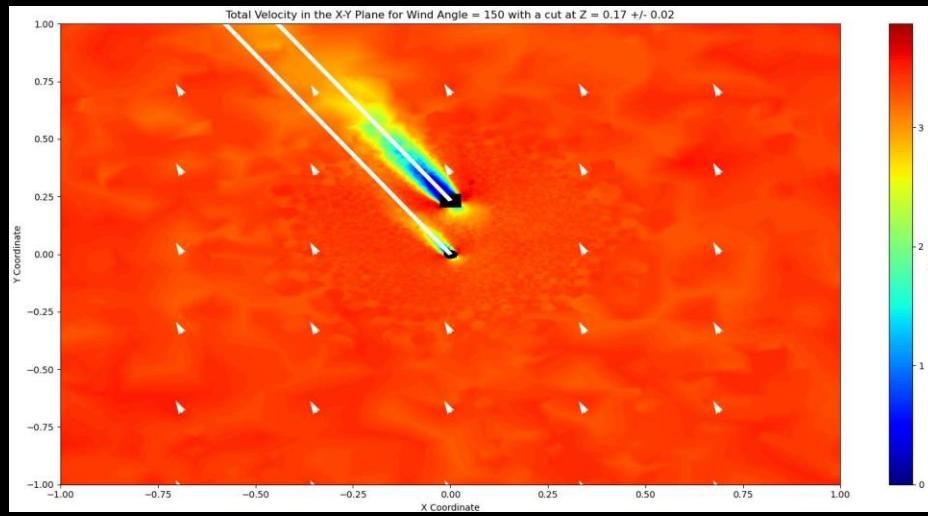


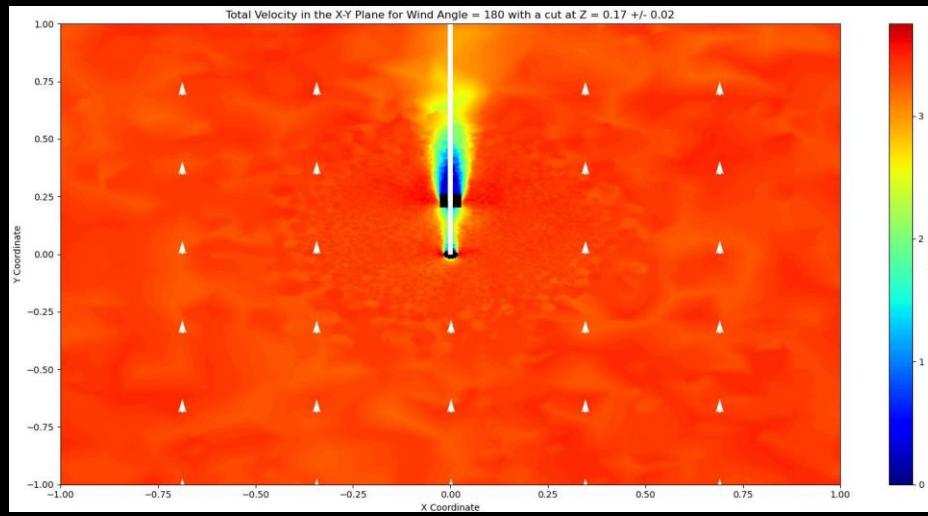


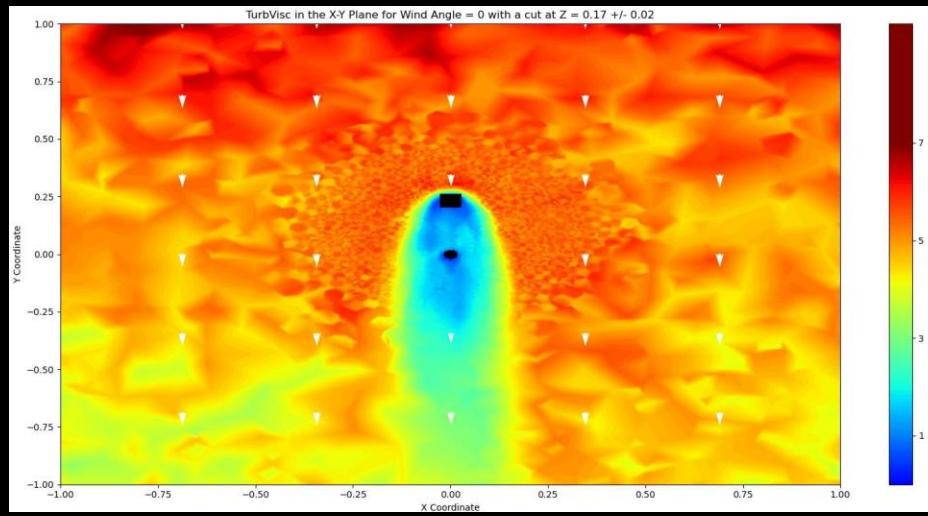


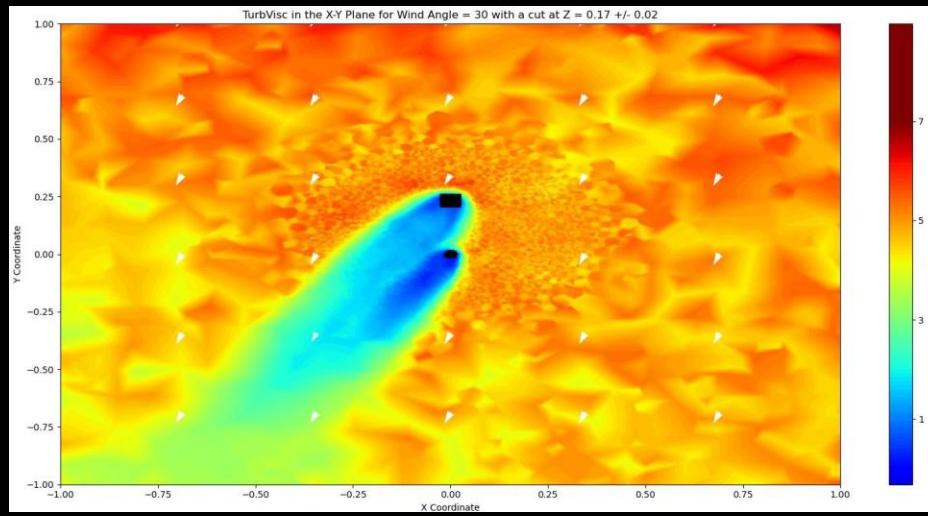


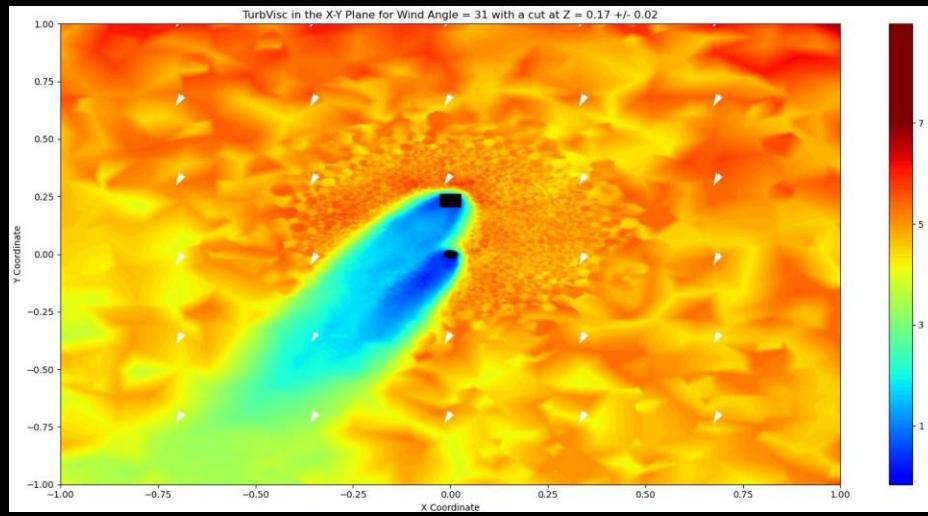


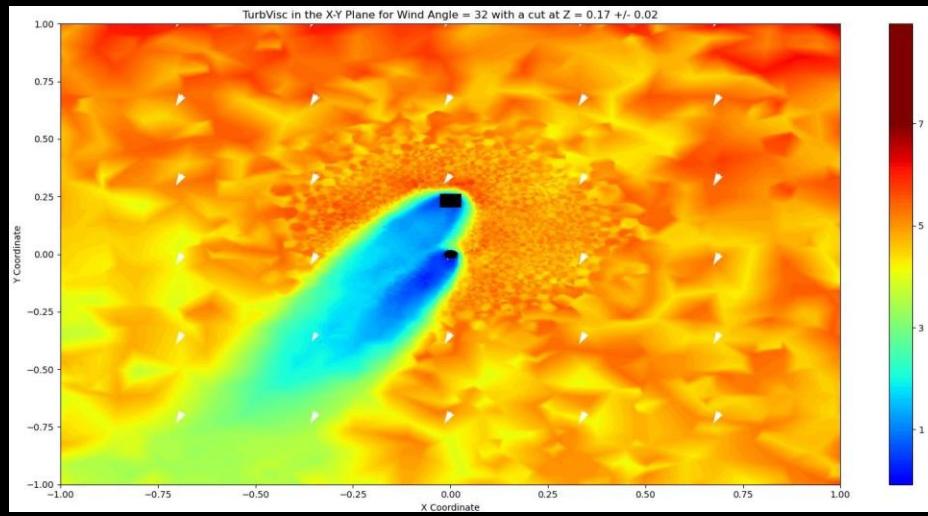


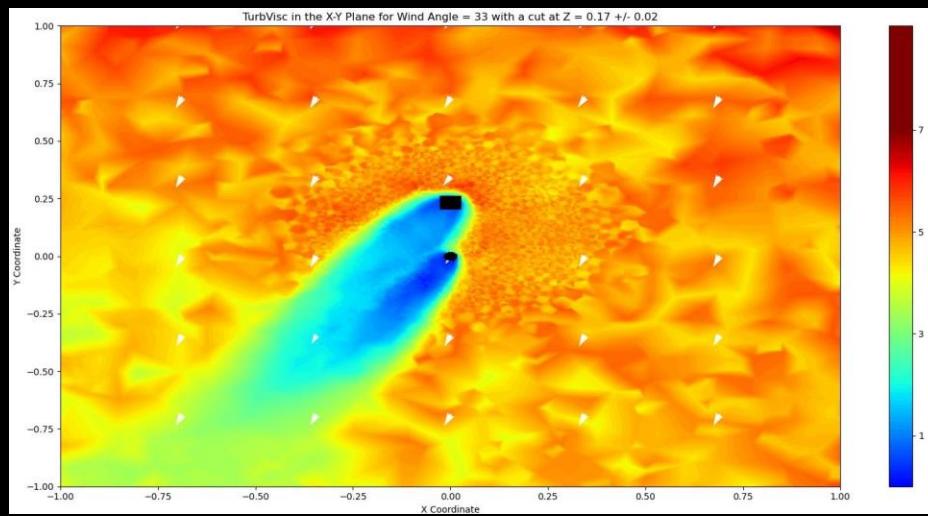


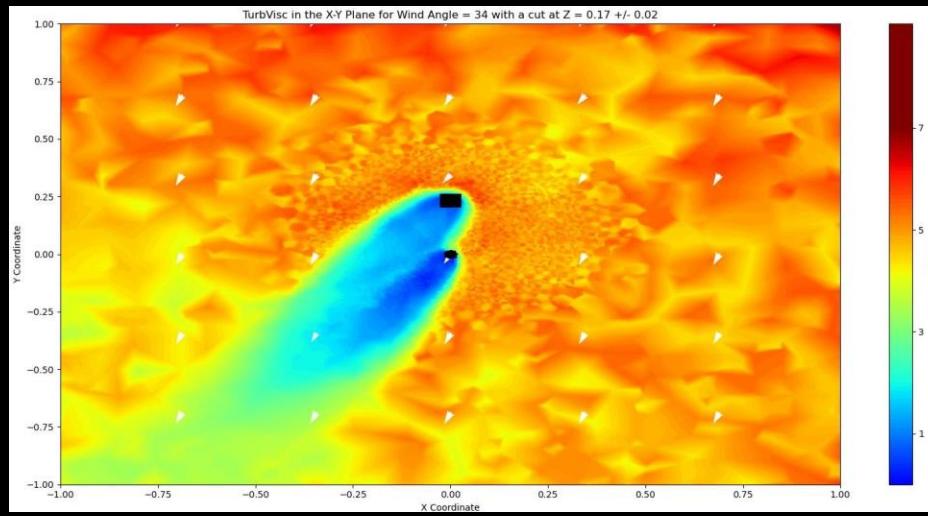


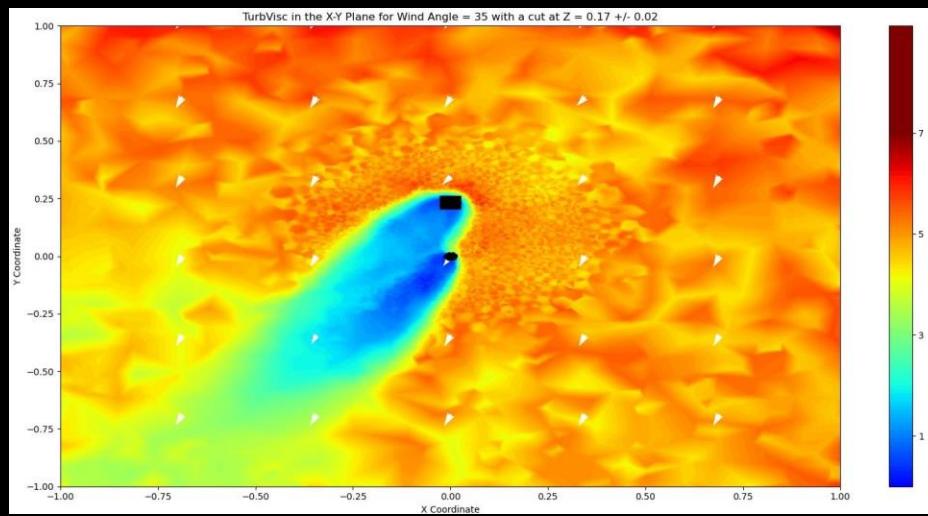


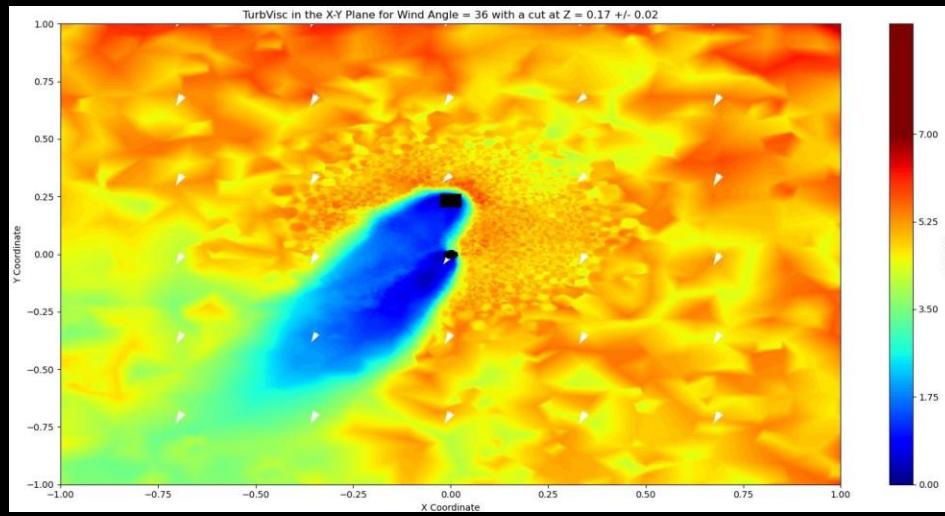


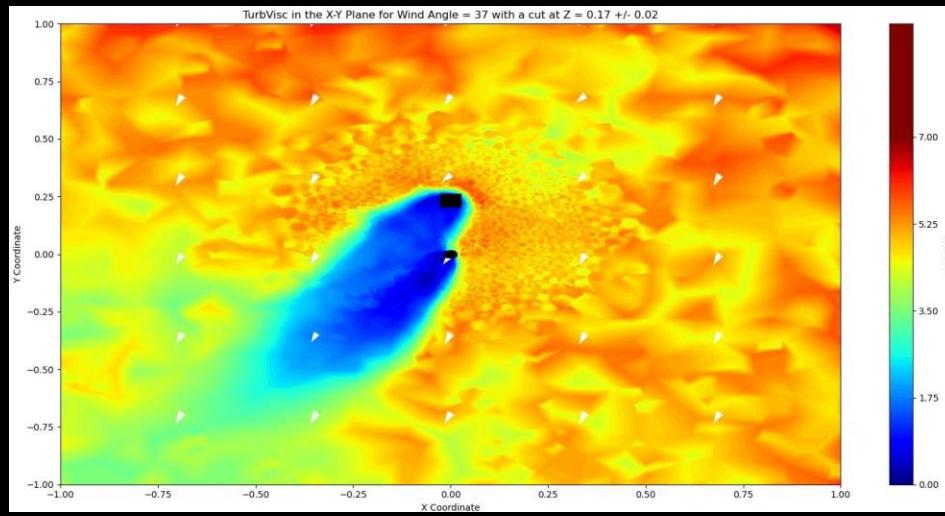


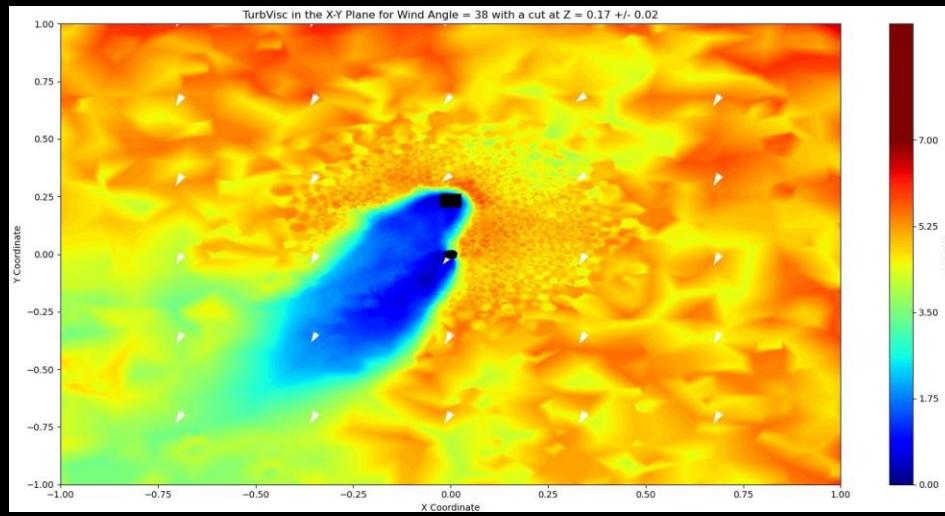


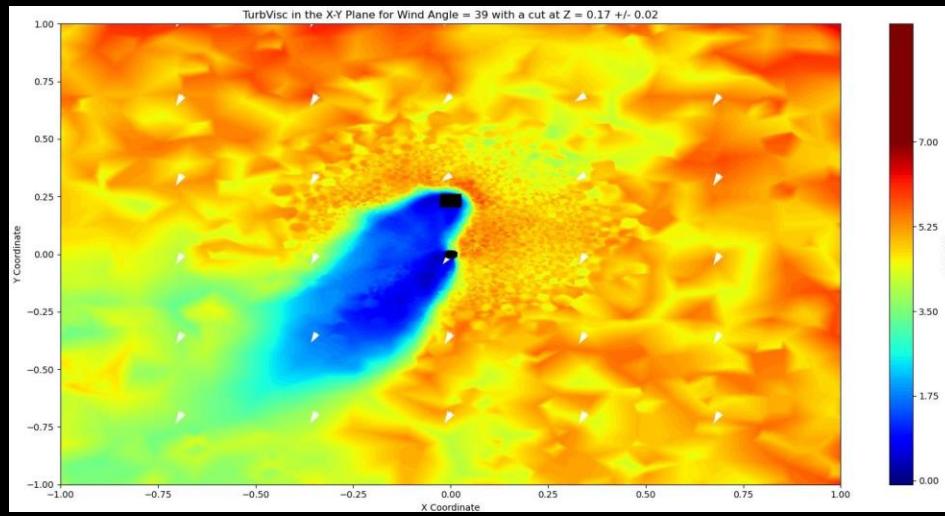


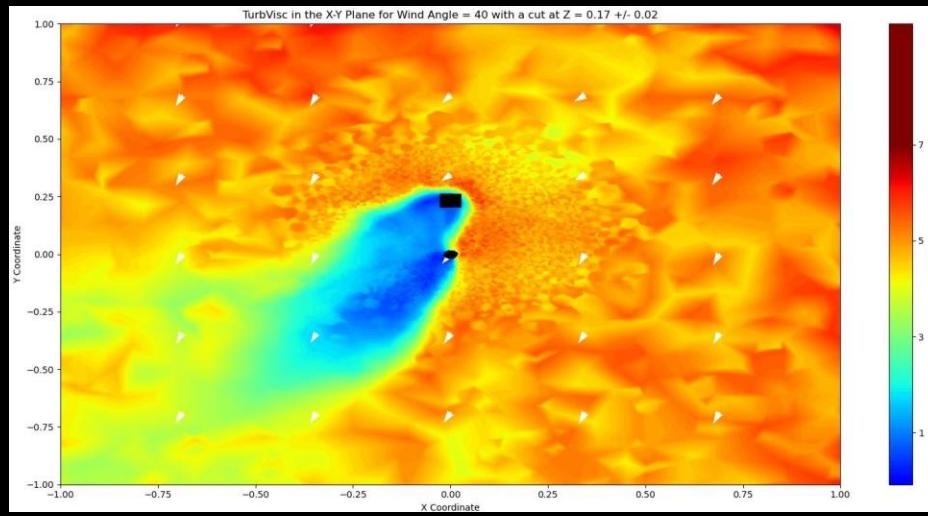


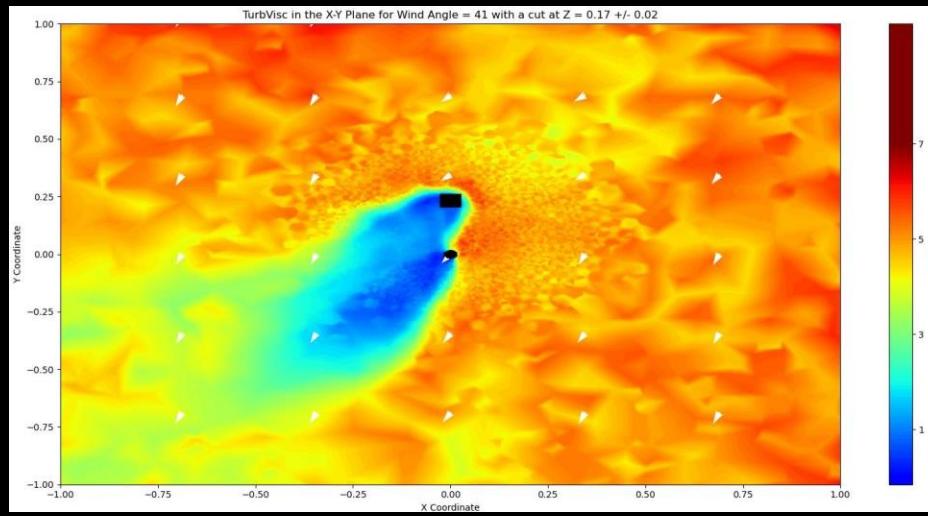


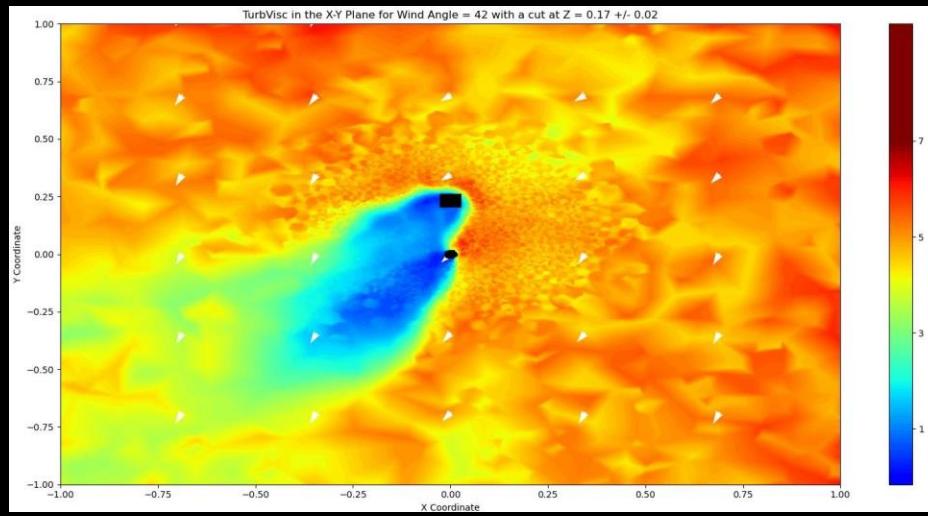


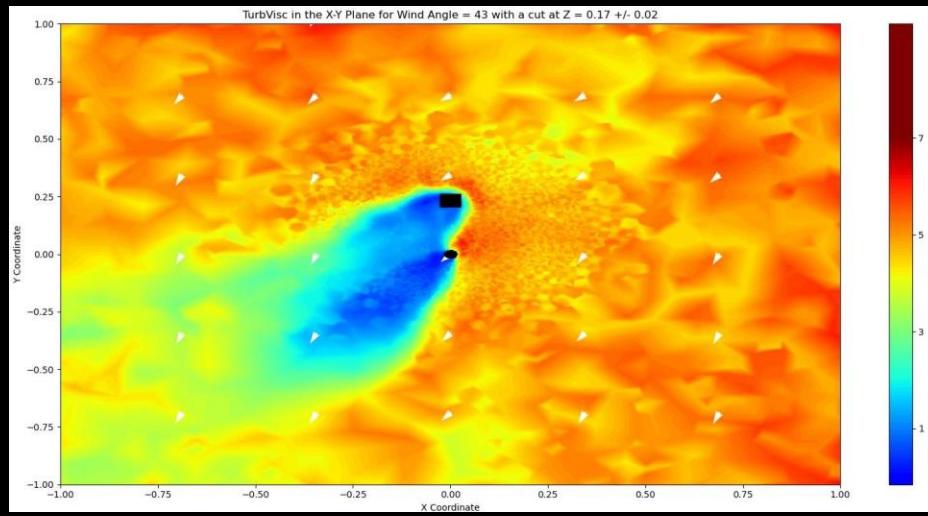


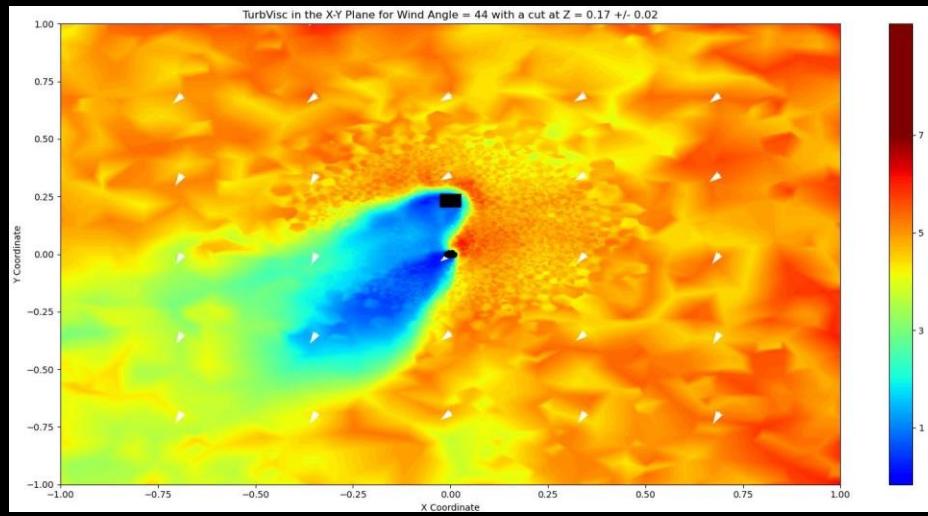


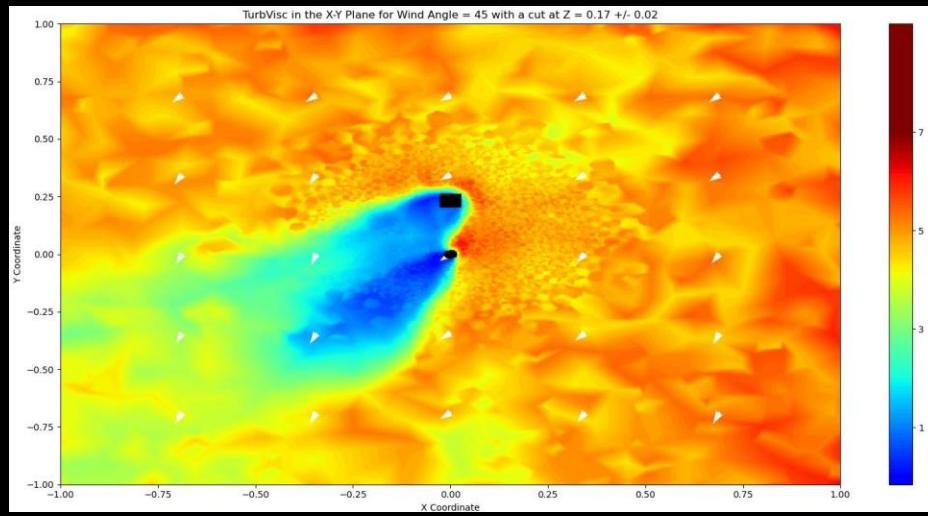


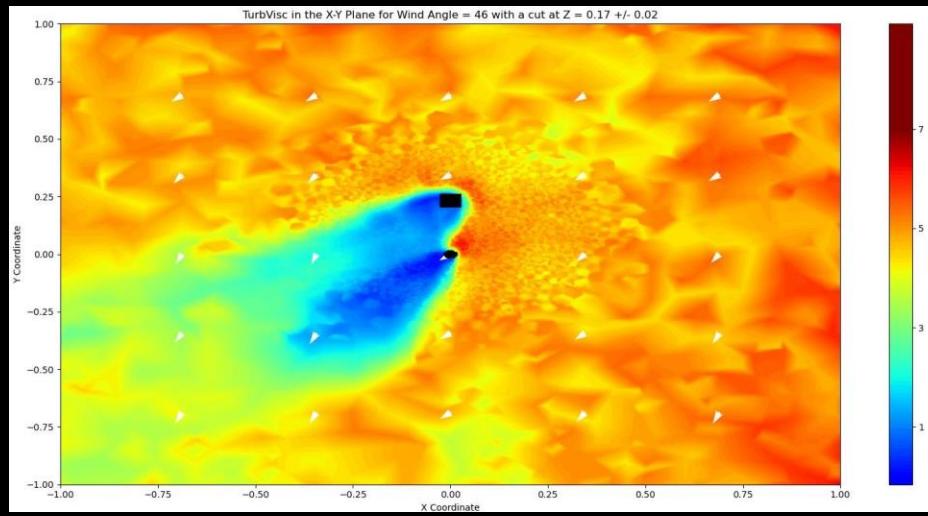


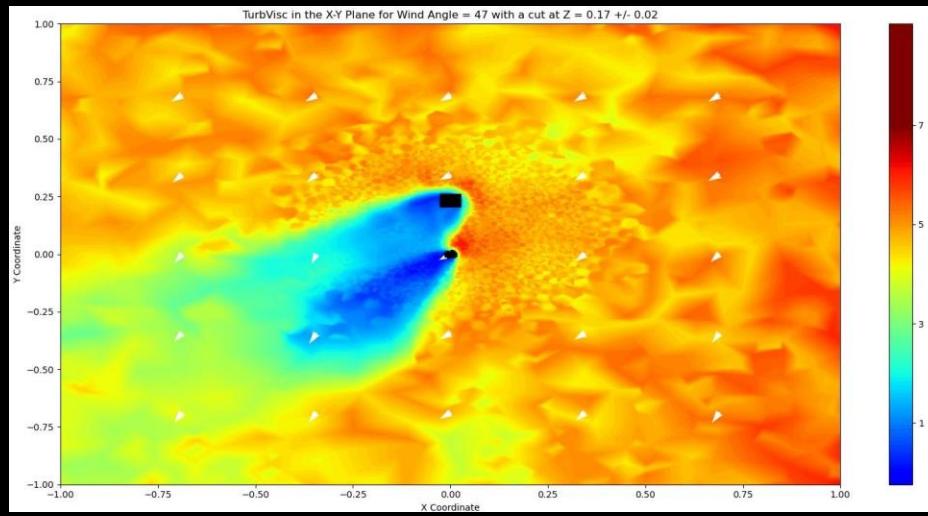


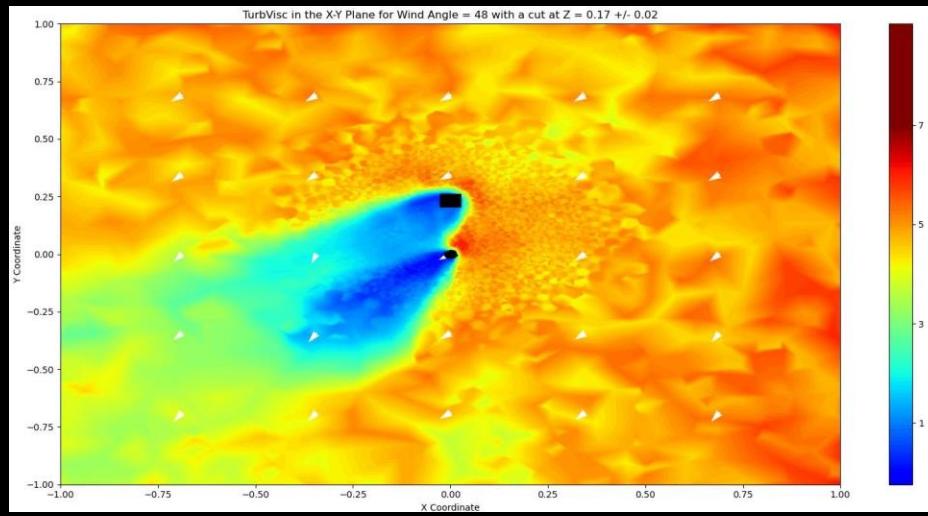


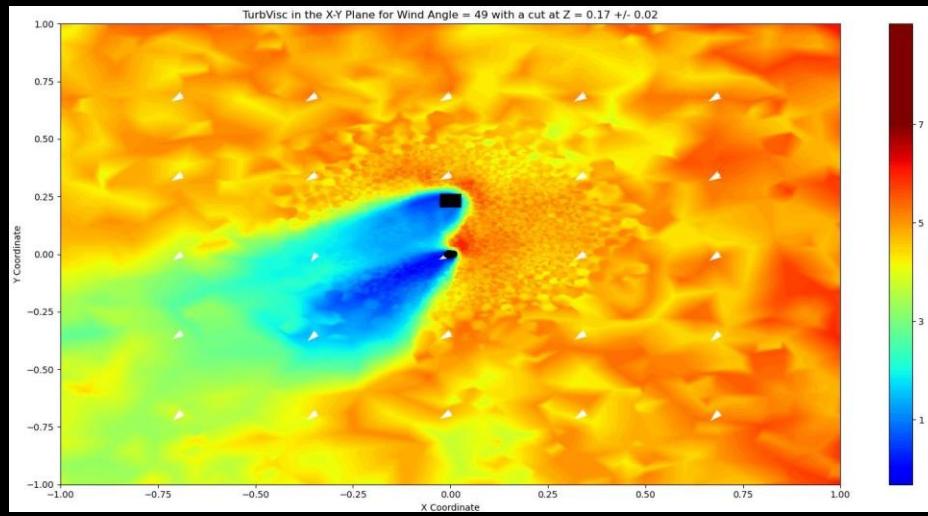


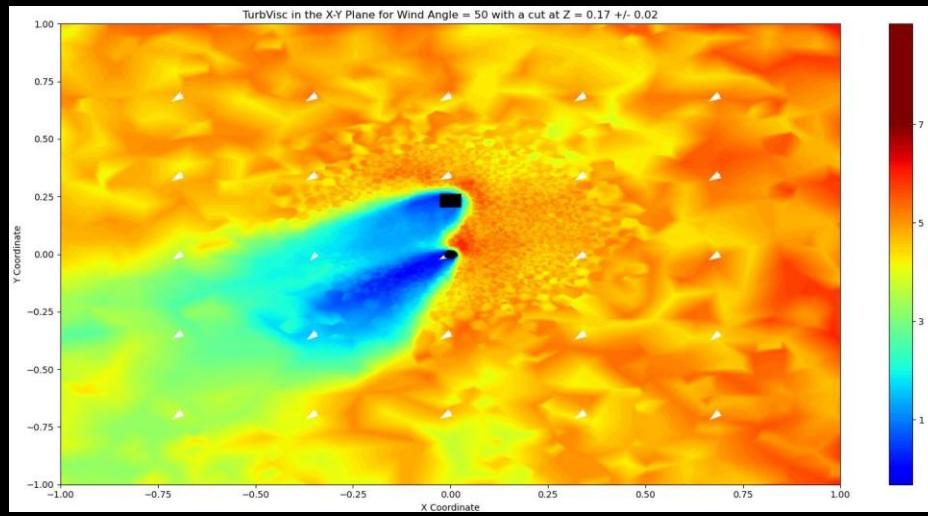


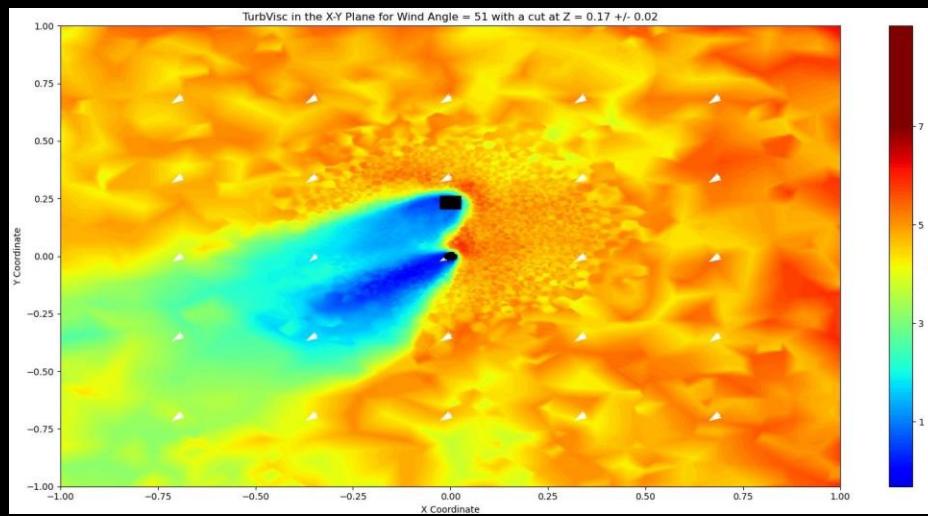


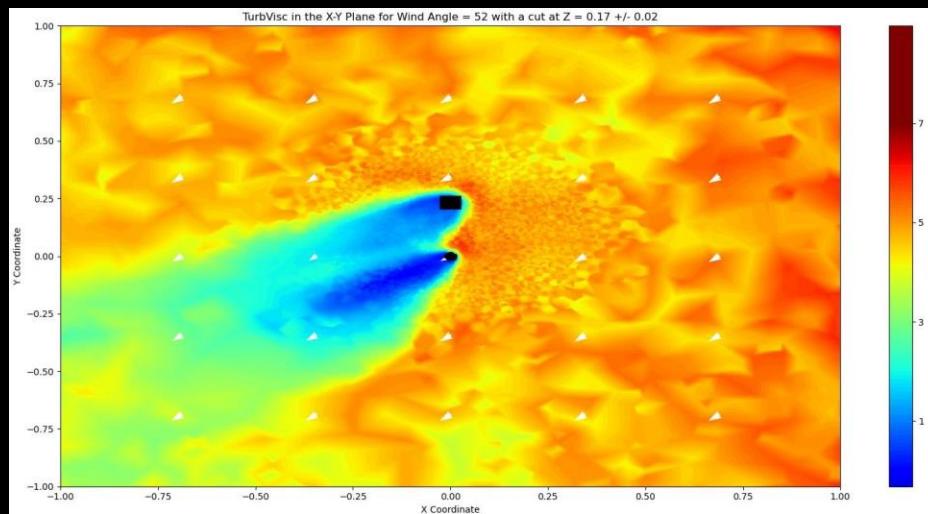


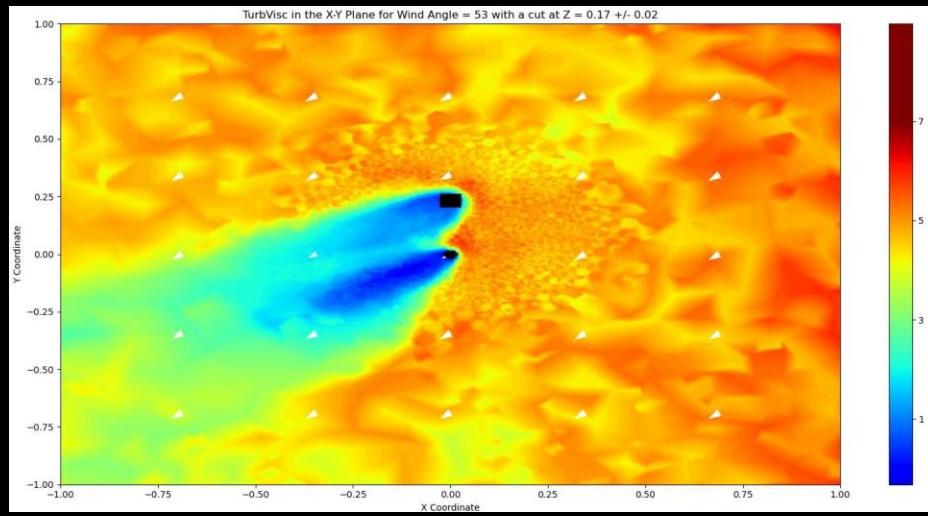


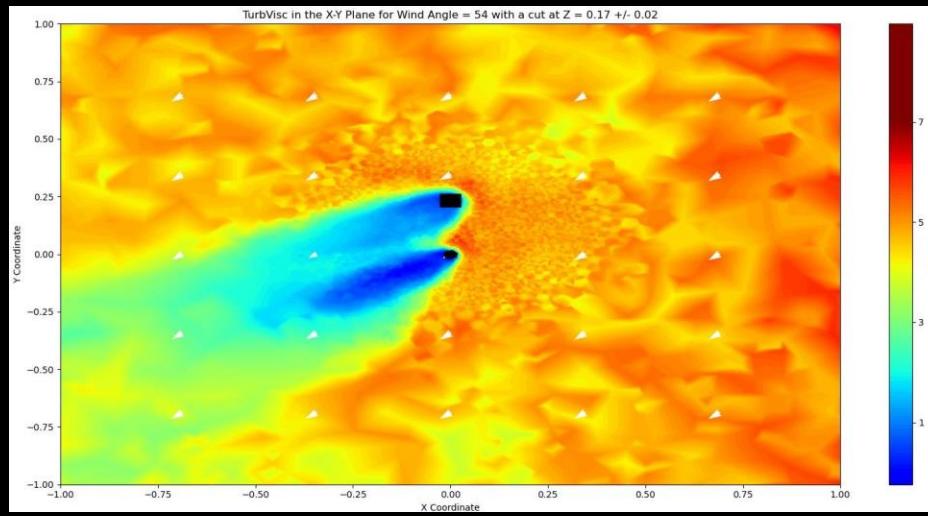


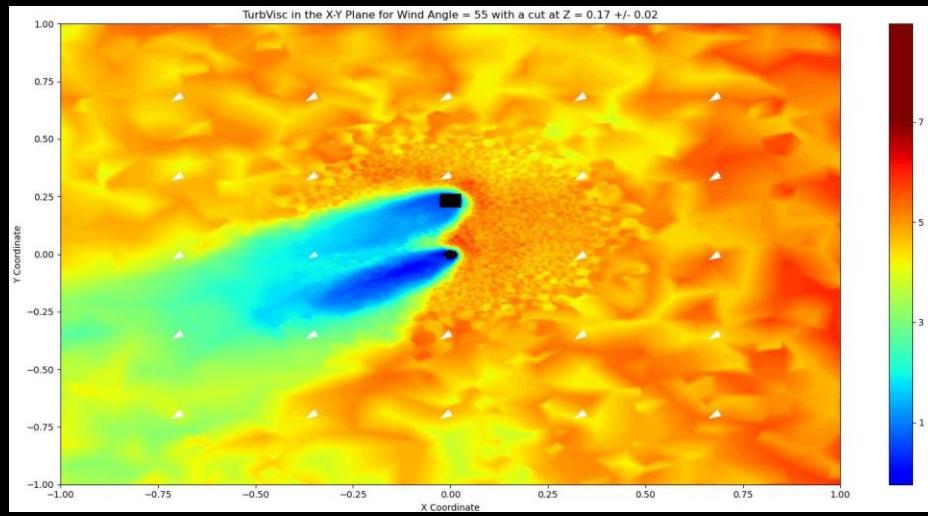


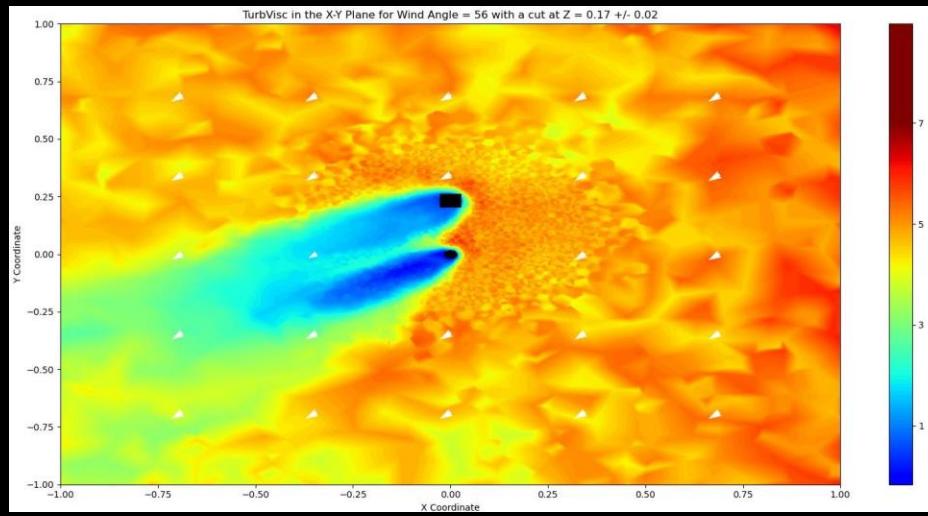


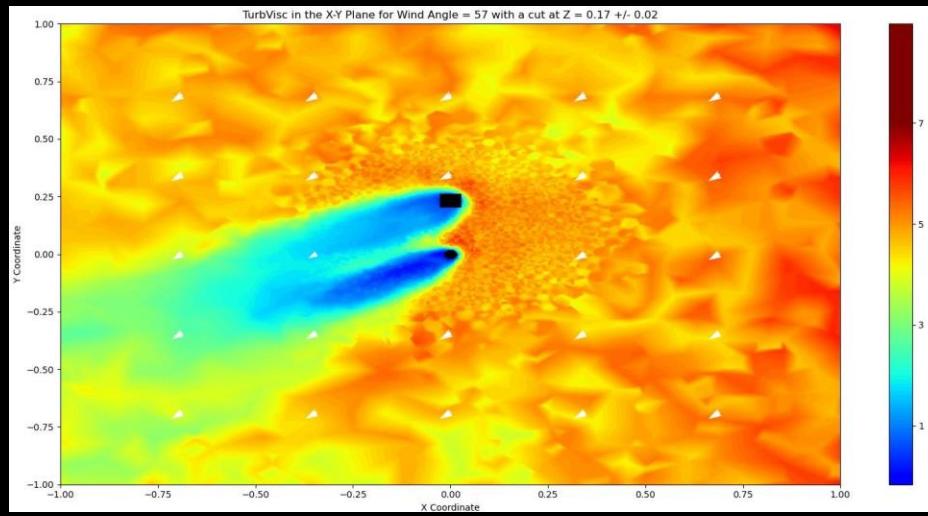


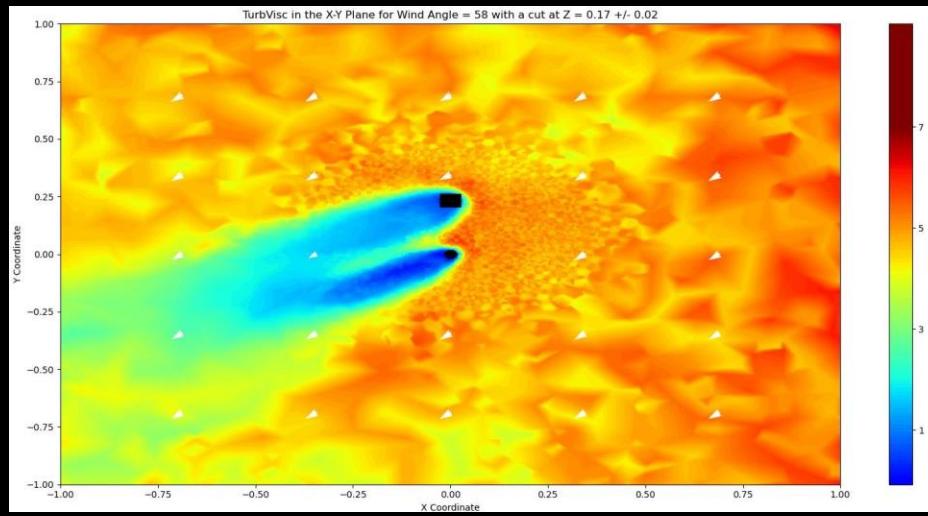


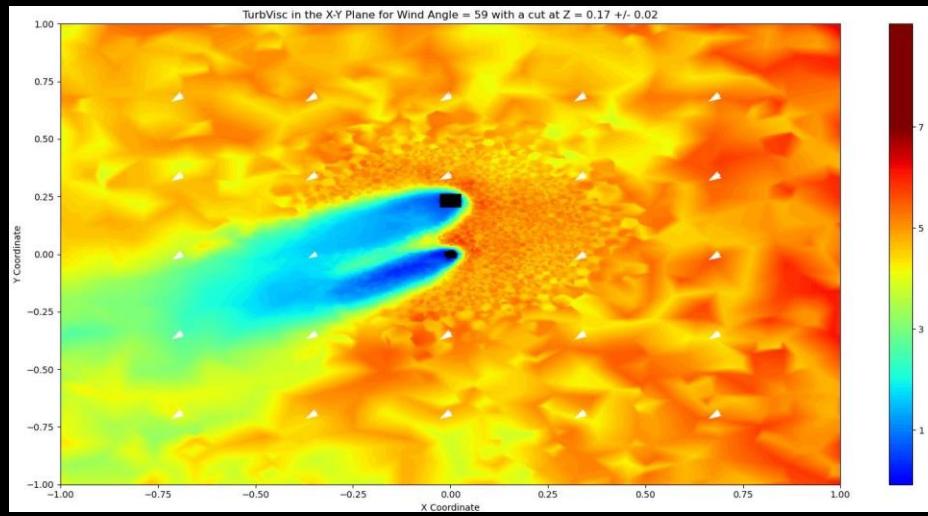


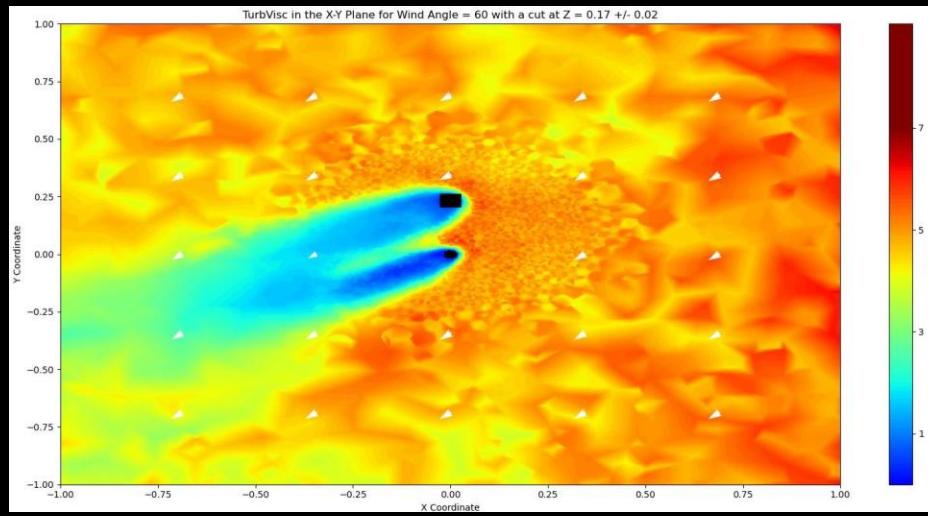


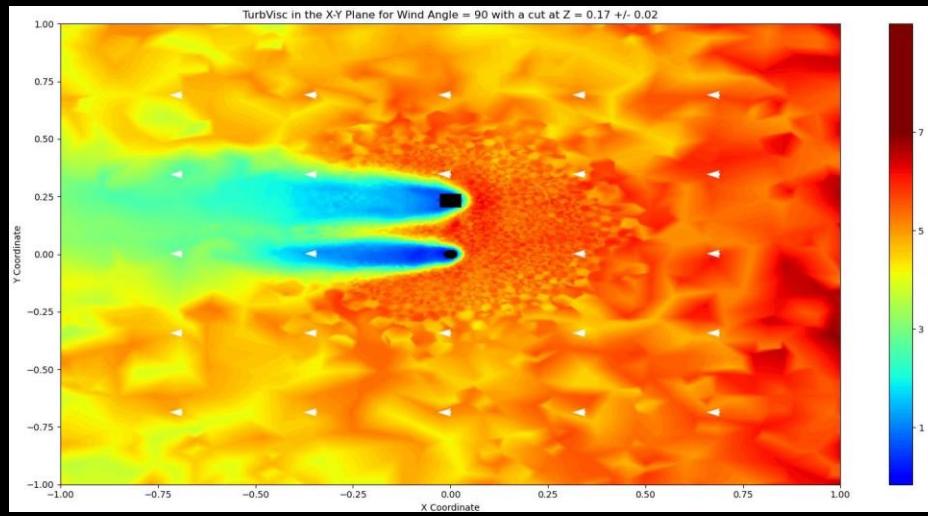


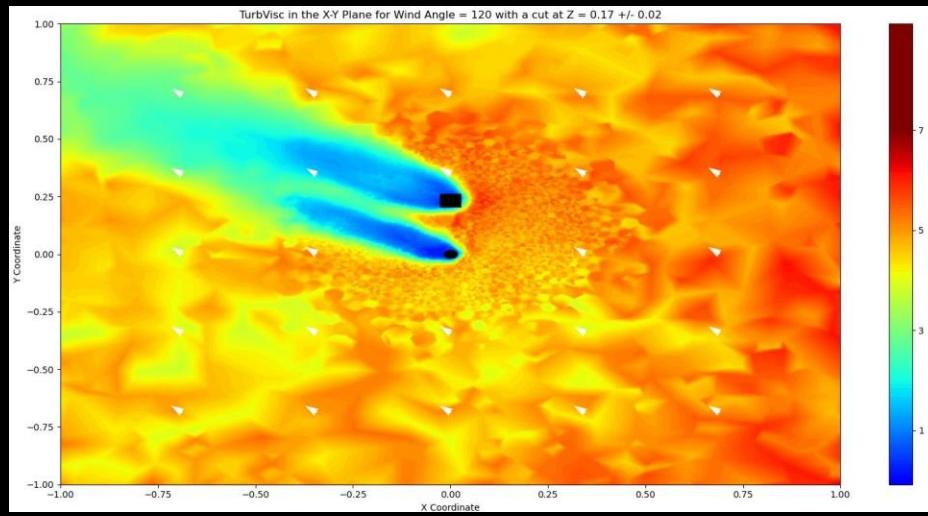


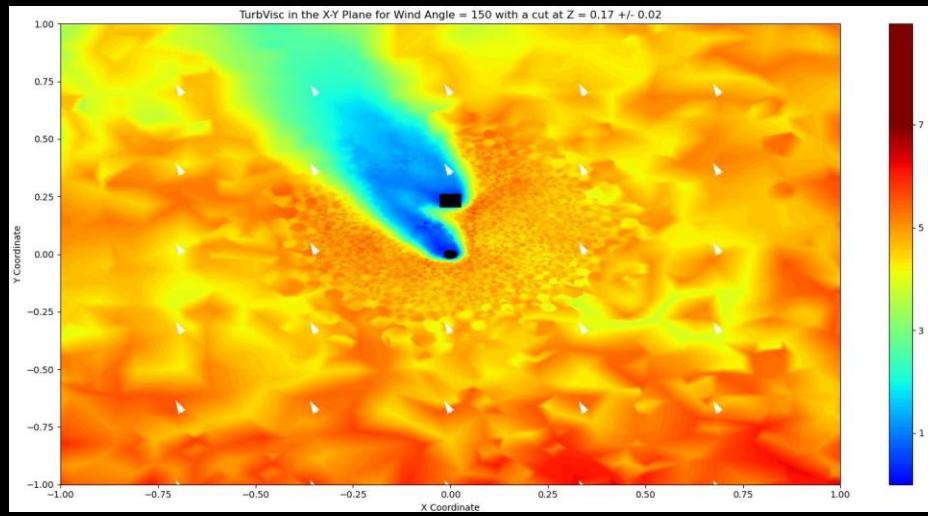


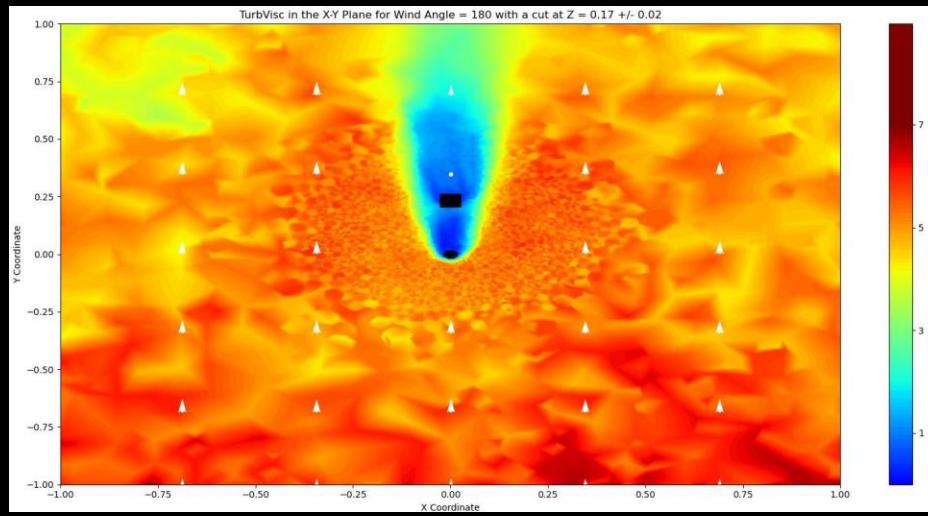












Progress so far - Data Loss + Cont Loss  
(WITHOUT Boundary Conditions imposed)  
(Adam Optimizer)

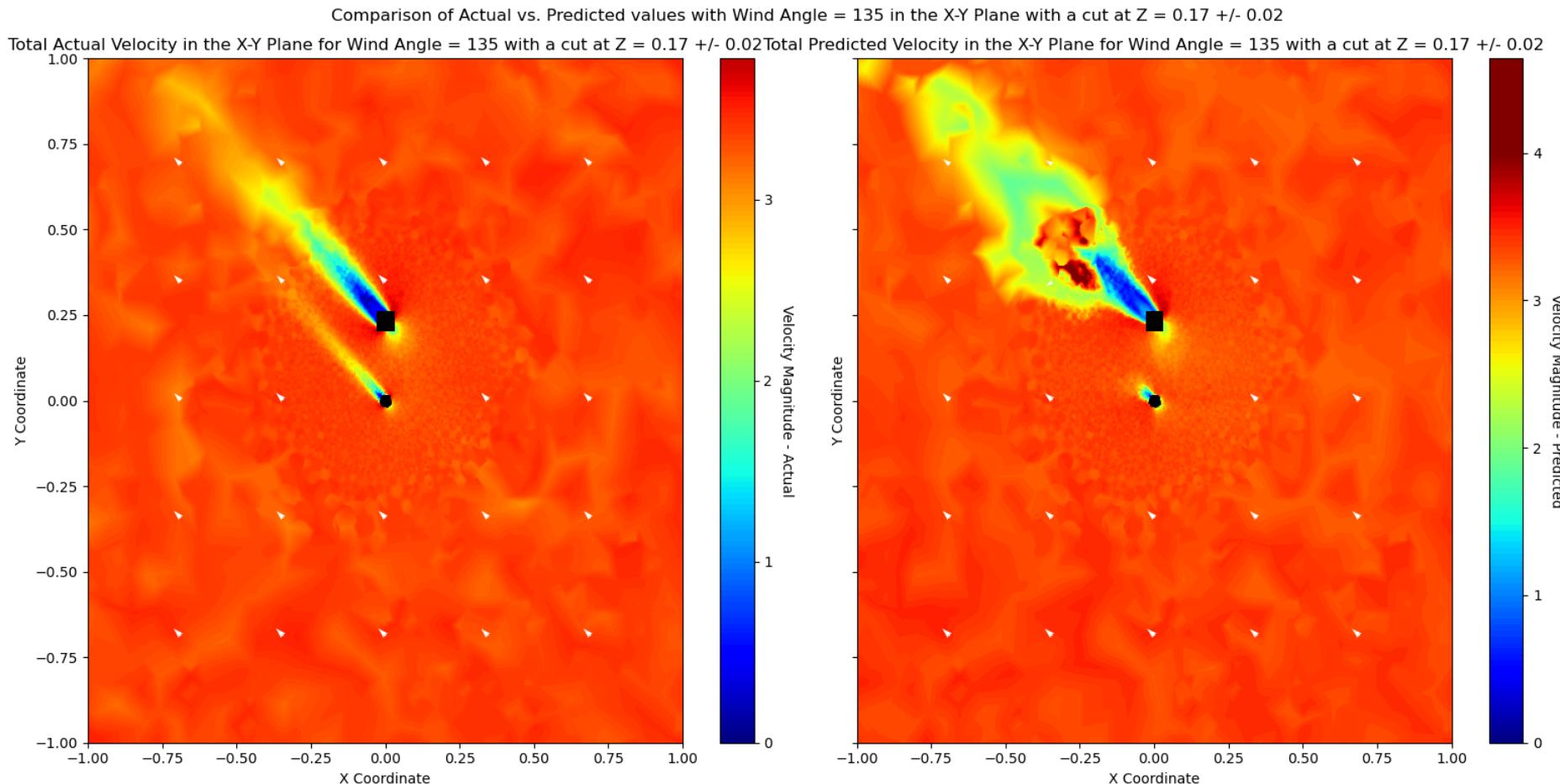
Threshold = 1E-5 (8810 Epochs, not completed), Google Colab

Scripts v3 – PREDICTING (135 DEG)

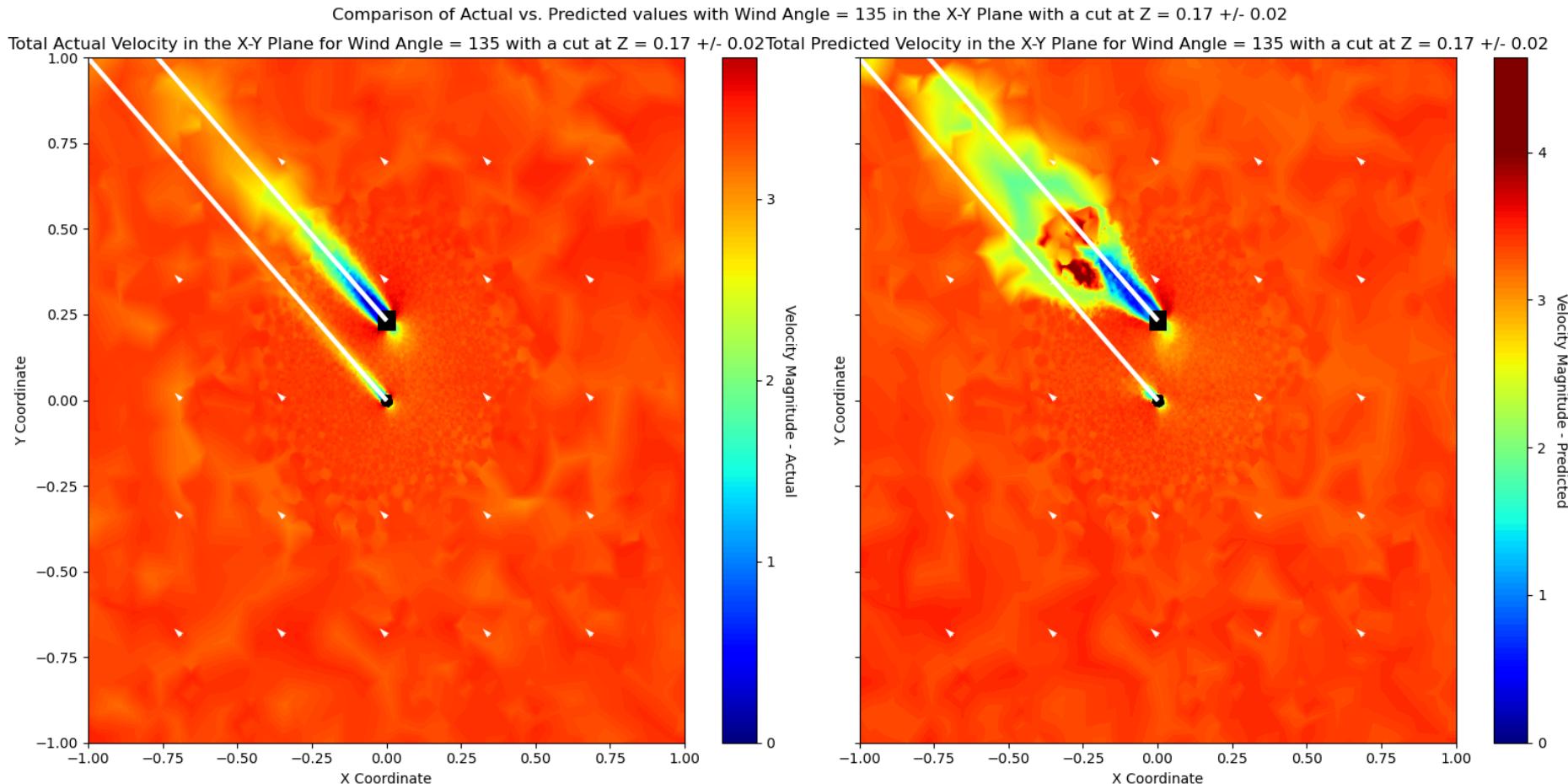
Progress so far - Data + Cont Loss (Adam Optimizer)  
Threshold = 1E-5 (8810 Epochs, so far...), Google Colab  
Predicting Results – Metrics (Angle = 135)

Variable	MSE	RMSE	MAE	R2
Pressure	0.685449654	0.827918869	0.528369796	0.594495436
Velocity:0	0.118347663	0.344016952	0.160197904	0.883884907
Velocity:1	0.144505396	0.380138654	0.194494929	0.859572314
Velocity:2	0.007320889	0.085562195	0.030433448	0.776215818
TurbVisc	0.327236887	0.572046228	0.431612503	0.997615436

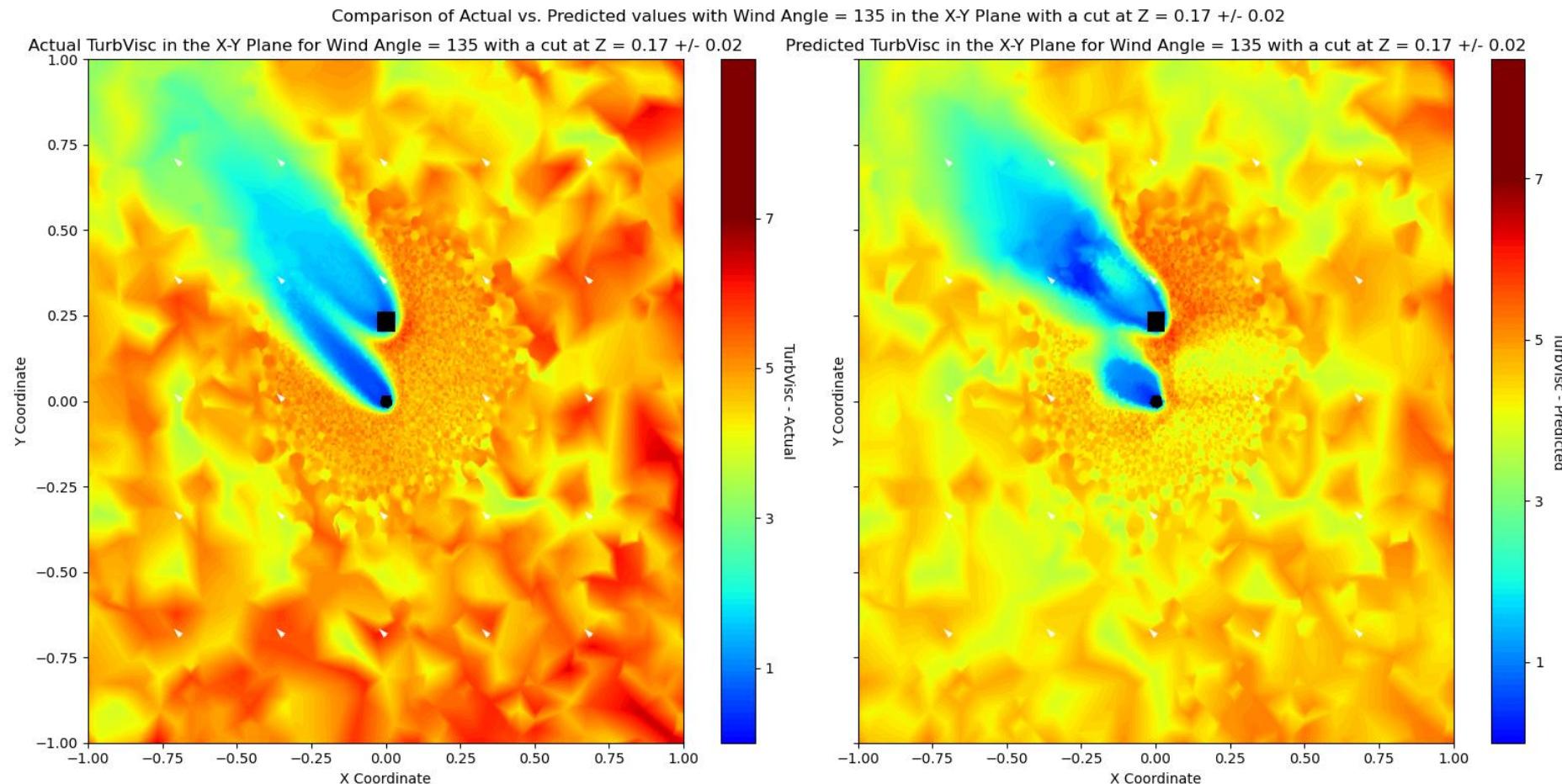
Progress so far - Data + Cont Loss (Adam Optimizer), Threshold = 1E-5 (8810 Epochs, so far...), Google Colab  
Predicting Results - X-Y Total Velocity Plot (Angle = 135)



Progress so far - Data + Cont Loss (Adam Optimizer), Threshold = 1E-5 (8810 Epochs, so far...), Google Colab  
Predicting Results - X-Y Total Velocity Plot (Angle = 135)

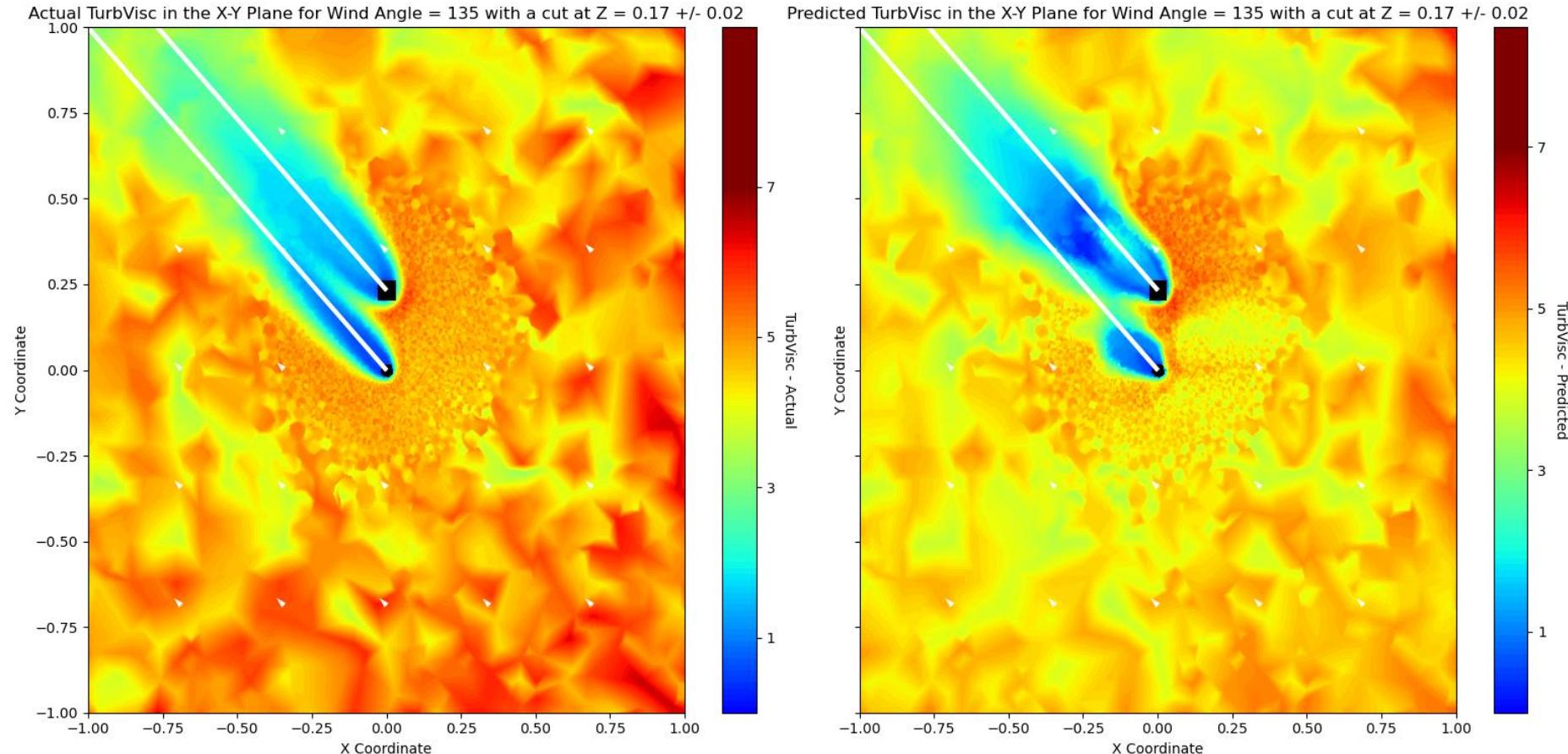


Progress so far - Data + Cont Loss (Adam Optimizer), Threshold = 1E-5 (8810 Epochs, so far...), Google Colab  
Predicting Results - X-Y TurbVisc Plot (Angle = 135)



Progress so far - Data + Cont Loss (Adam Optimizer), Threshold = 1E-5 (8810 Epochs, so far...), Google Colab  
Predicting Results - X-Y TurbVisc Plot (Angle = 135)

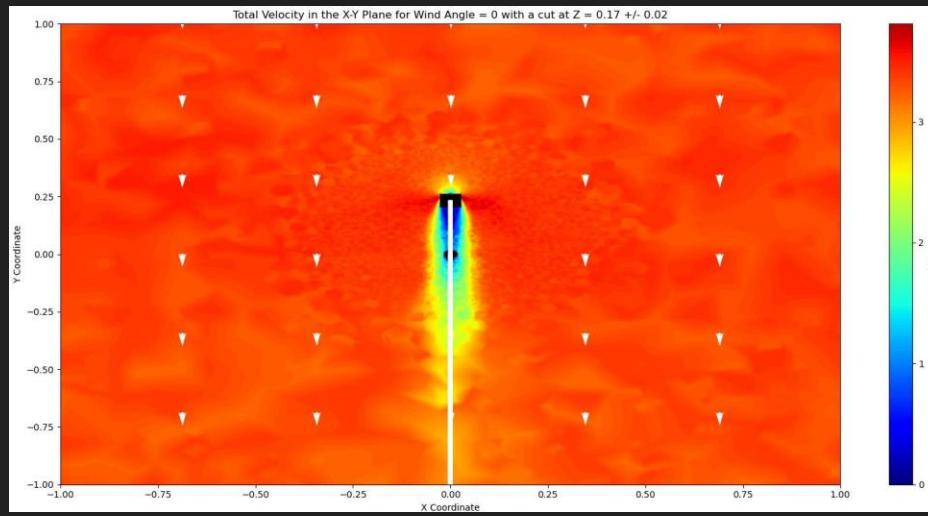
### Comparison of Actual vs. Predicted values with Wind Angle = 135 in the X-Y Plane with a cut at Z = 0.17 +/- 0.02

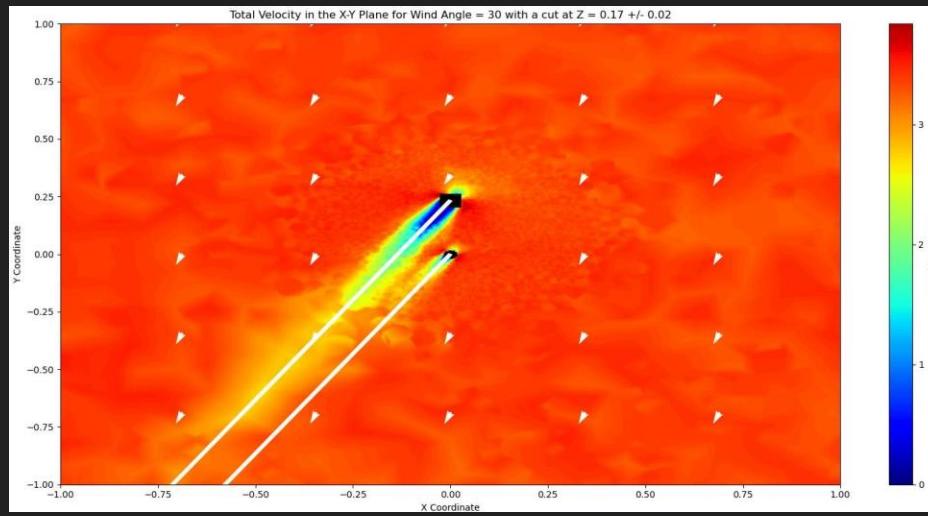


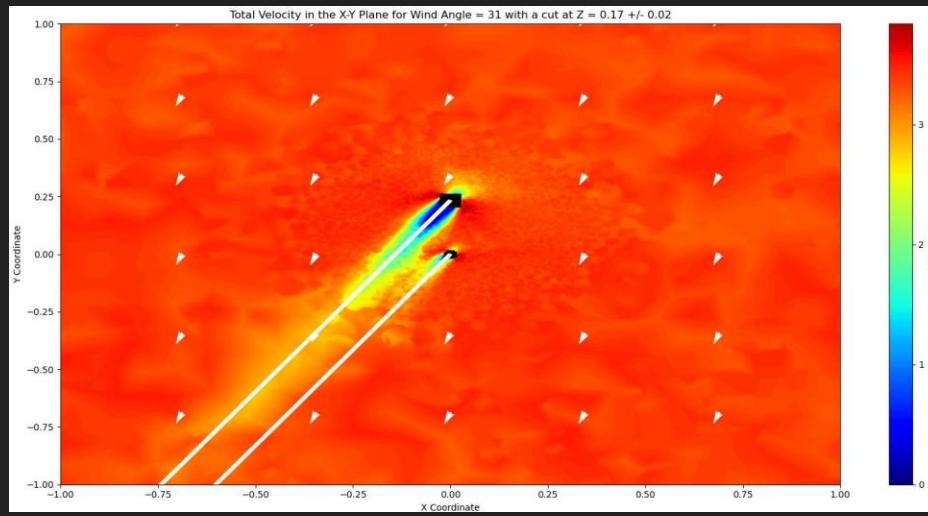
# Progress so far - Data Loss + Cont Loss (WITHOUT Boundary Conditions imposed) (Adam Optimizer)

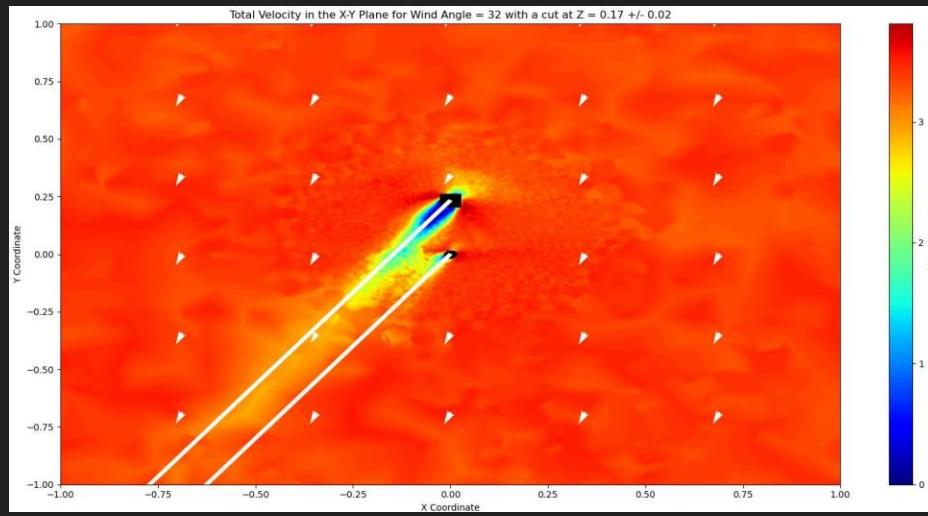
Threshold = 1E-5 (8810 Epochs, not completed), Google Colab

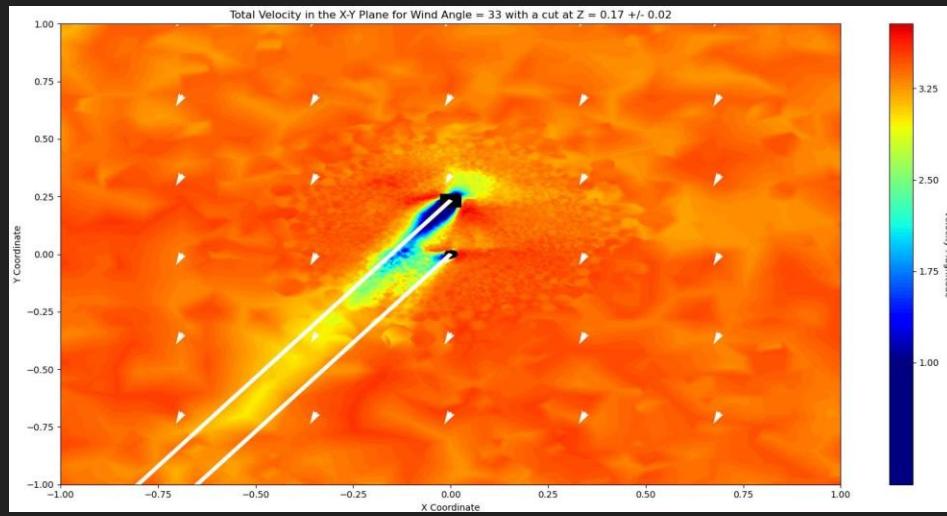
Scripts v3 – Plotting Any Angle

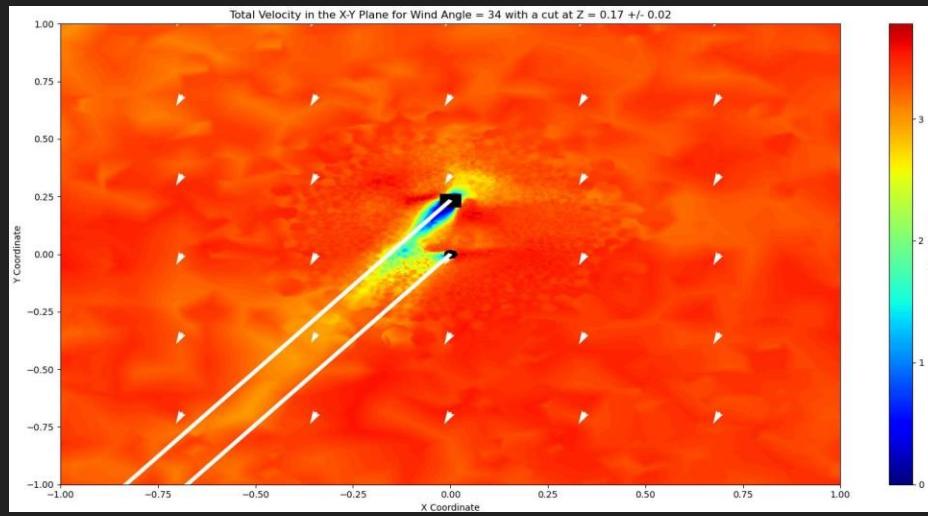


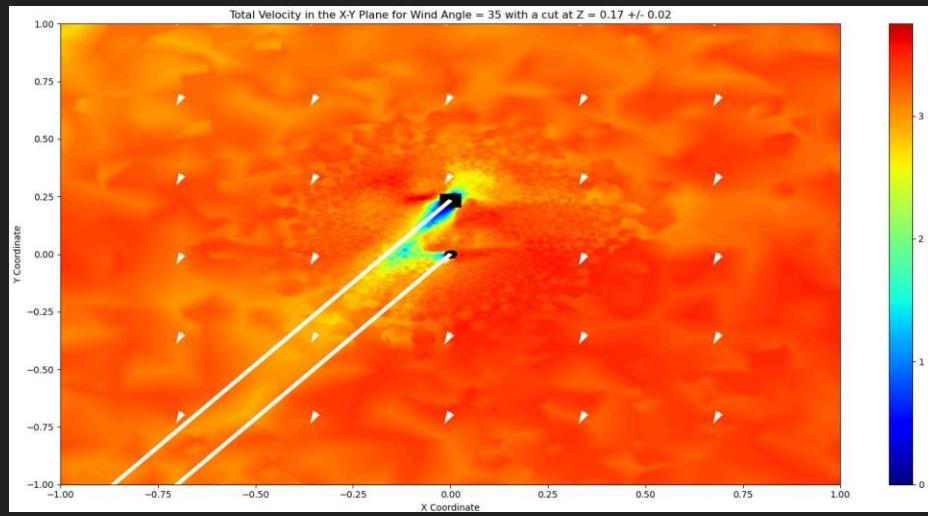


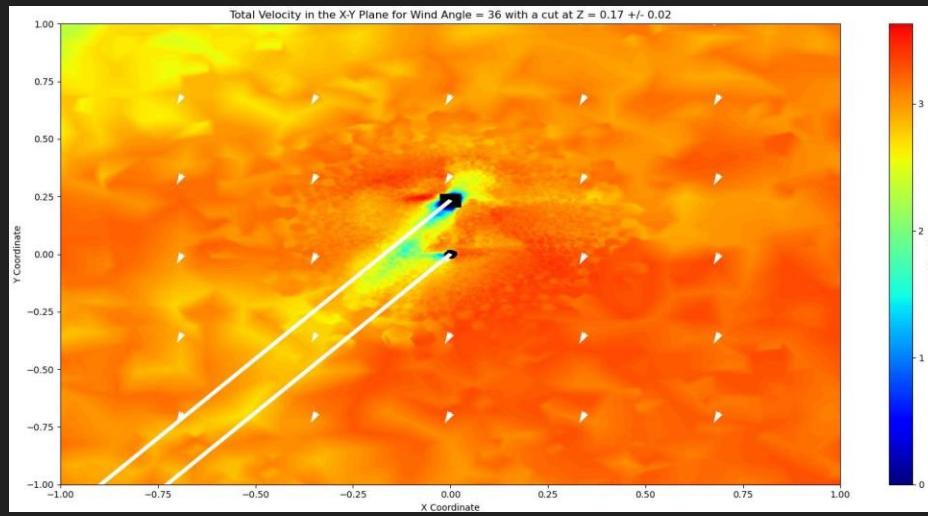


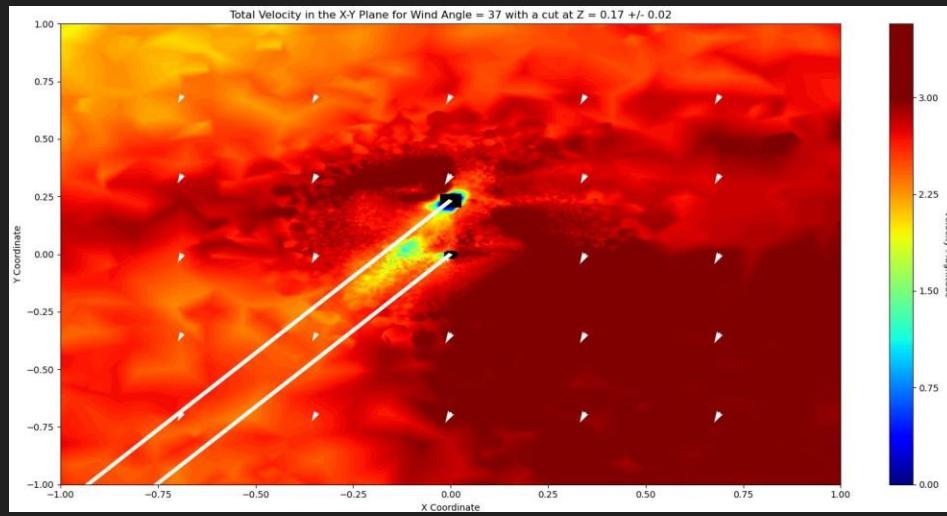


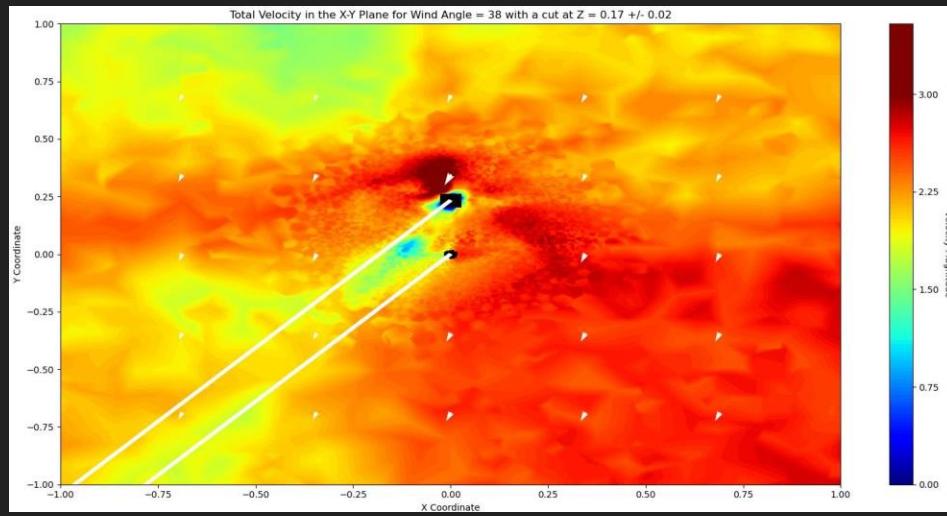


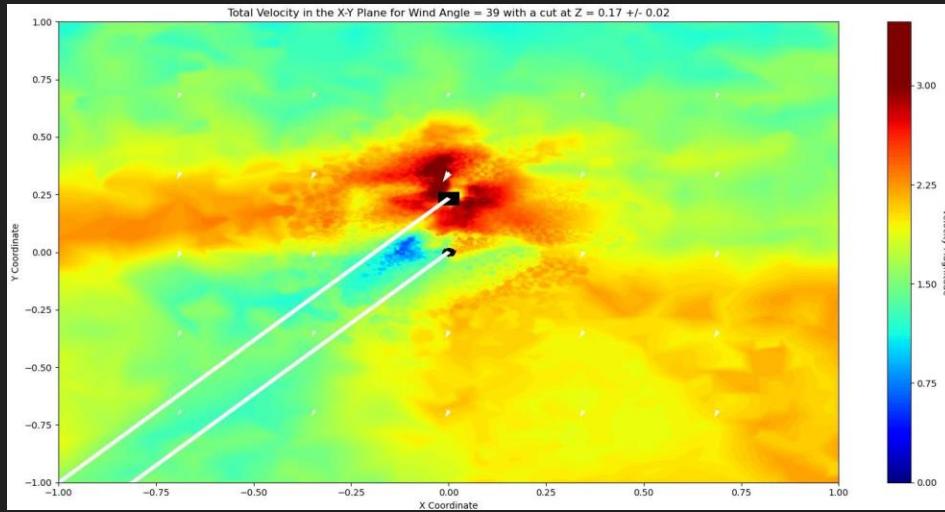


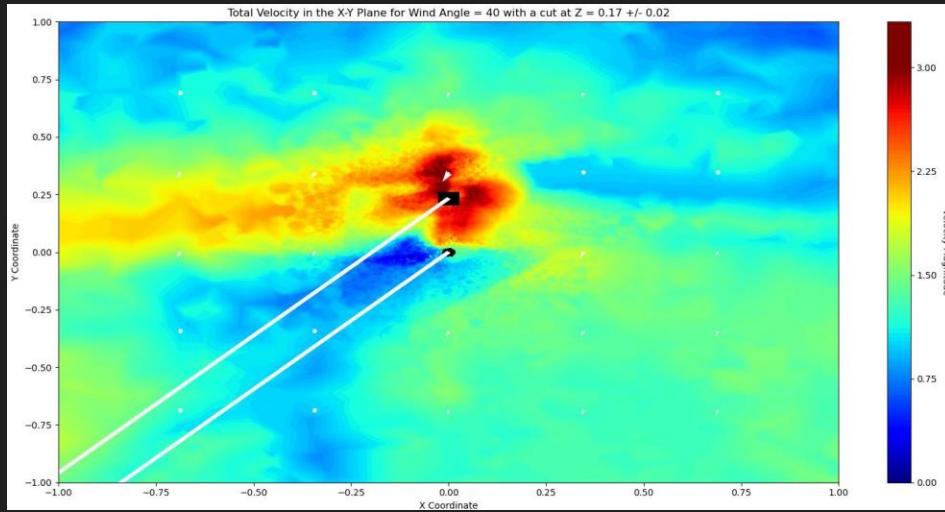


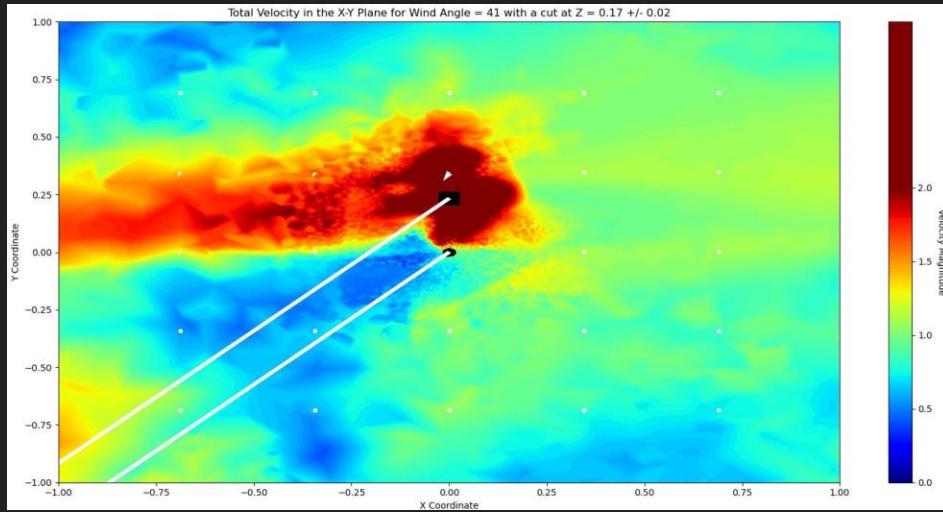


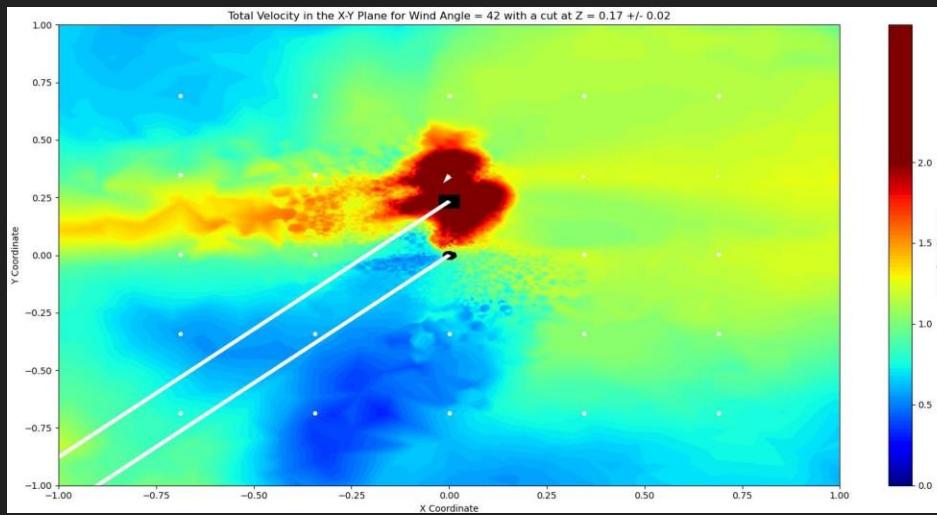


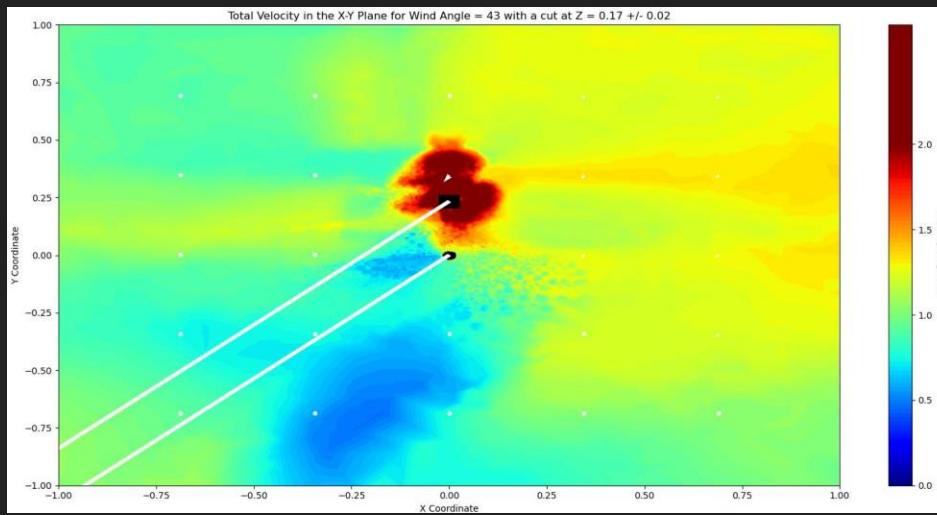


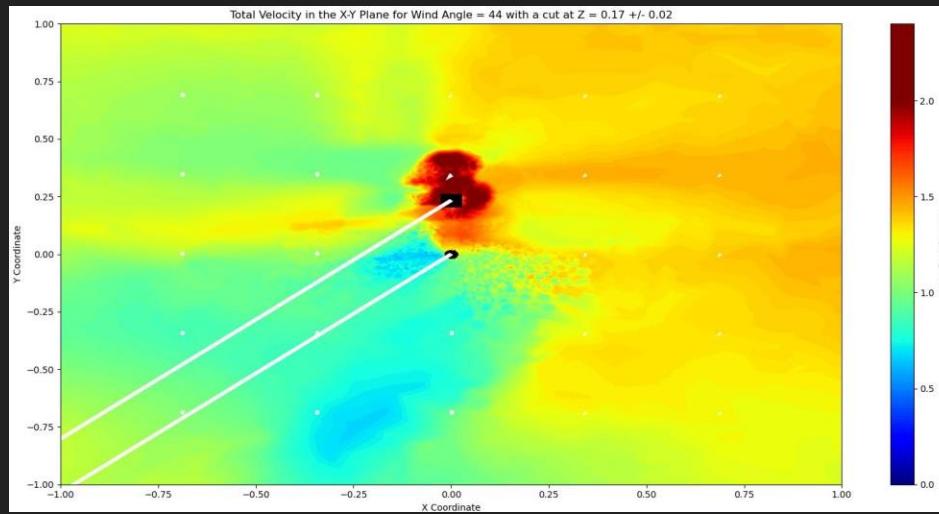


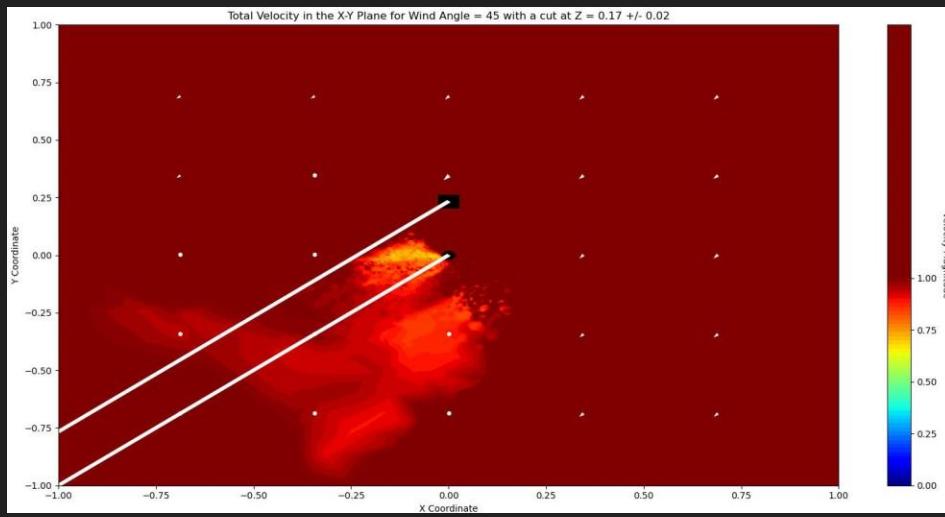


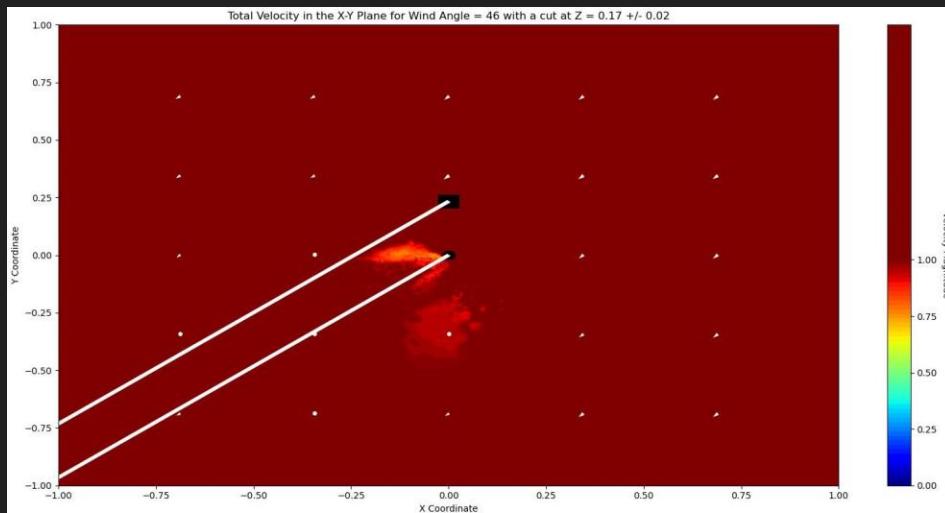


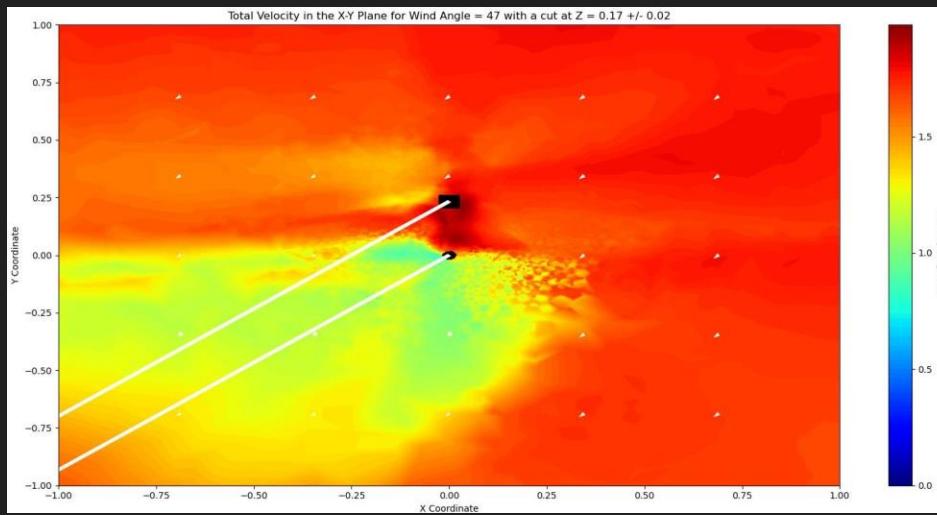


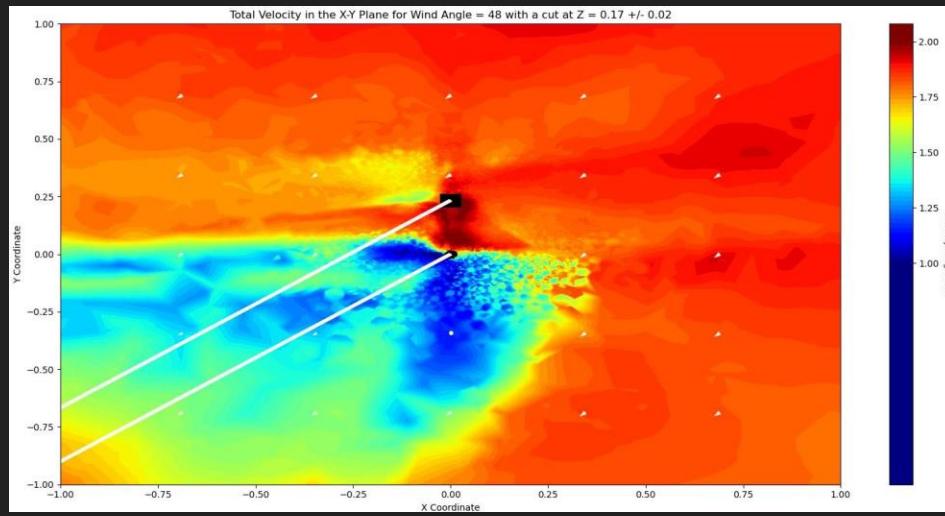


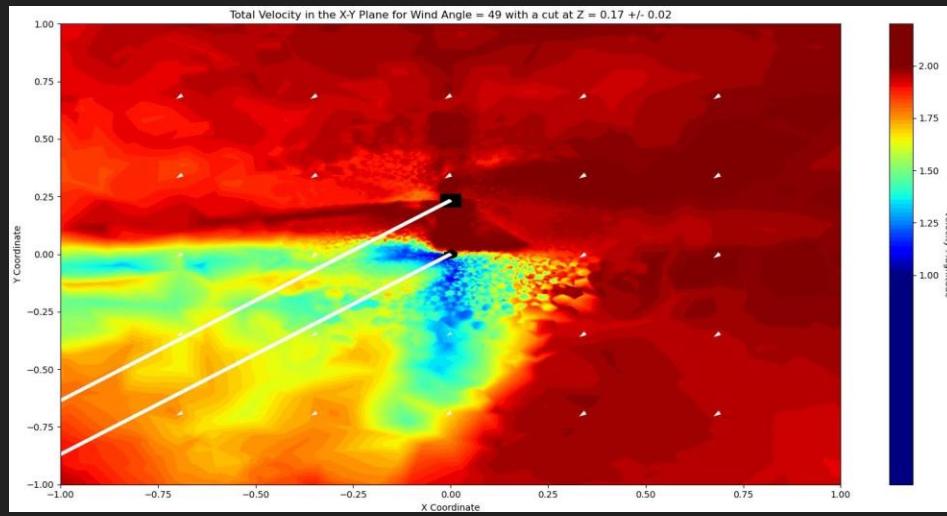


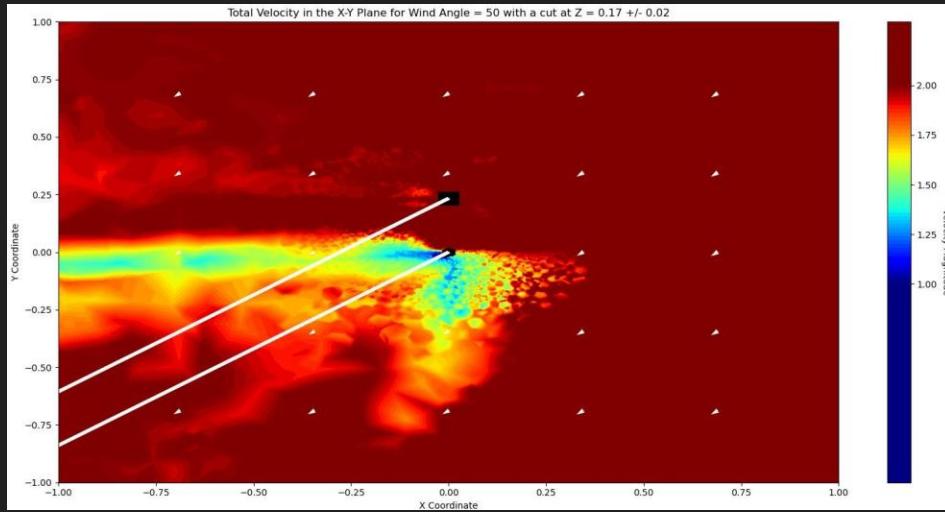


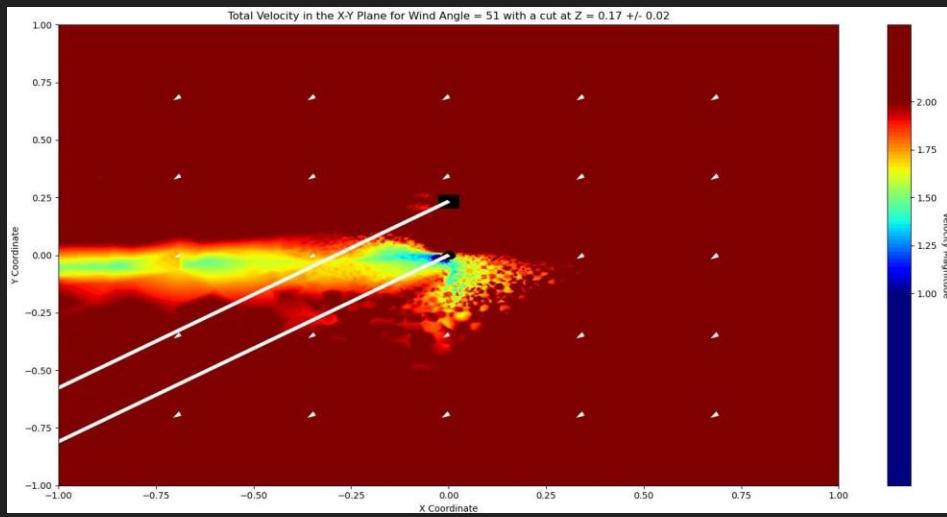


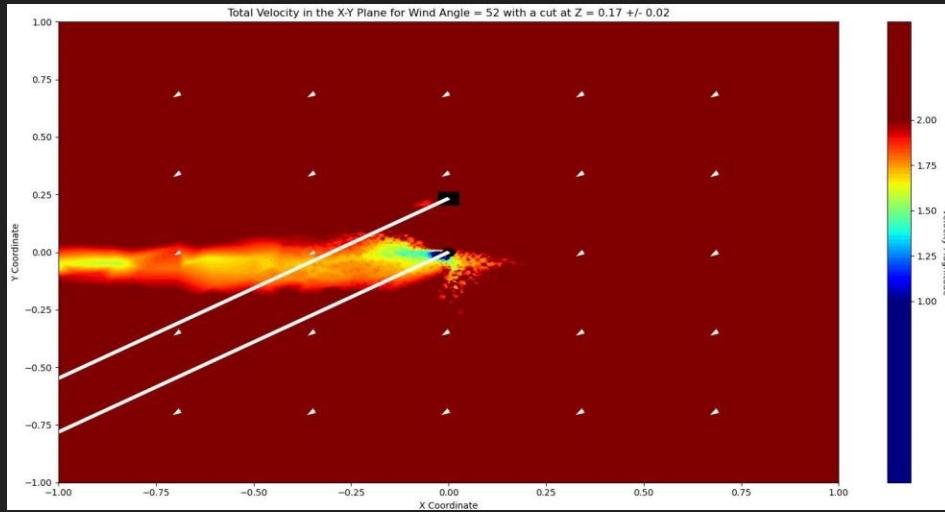


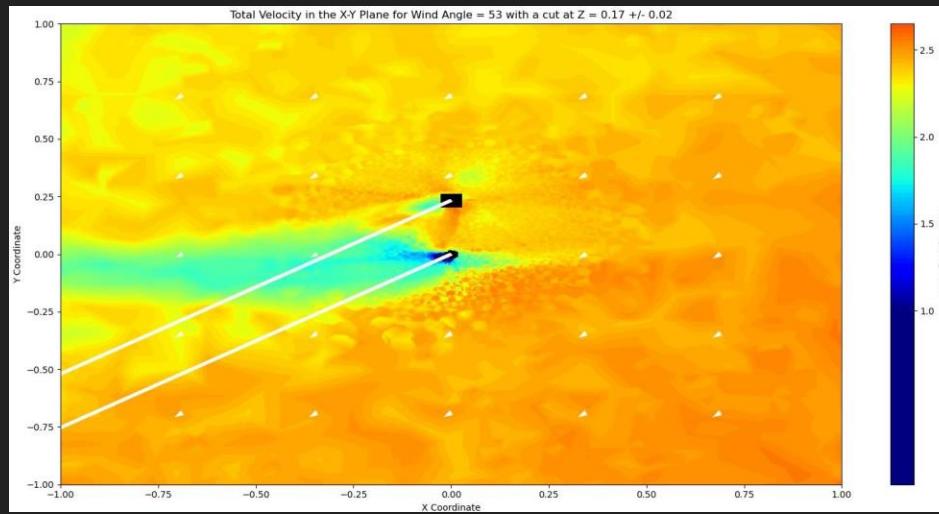


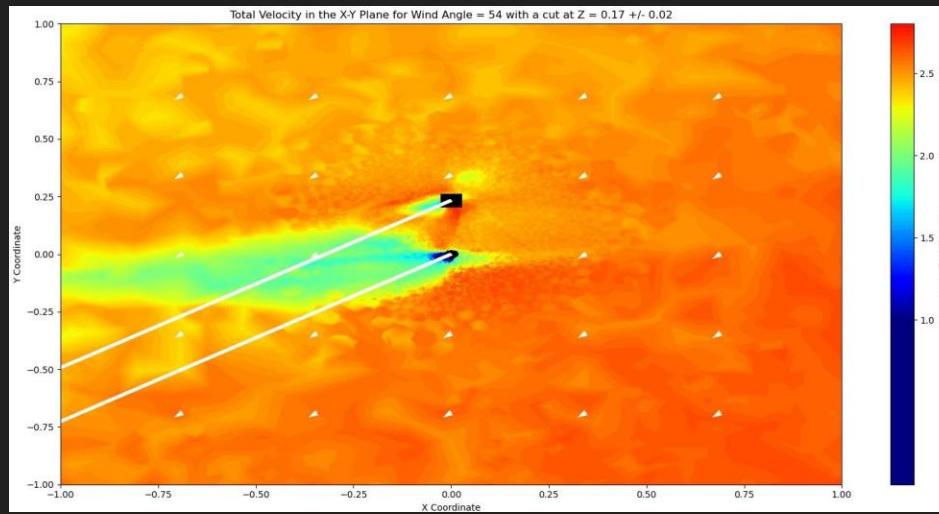


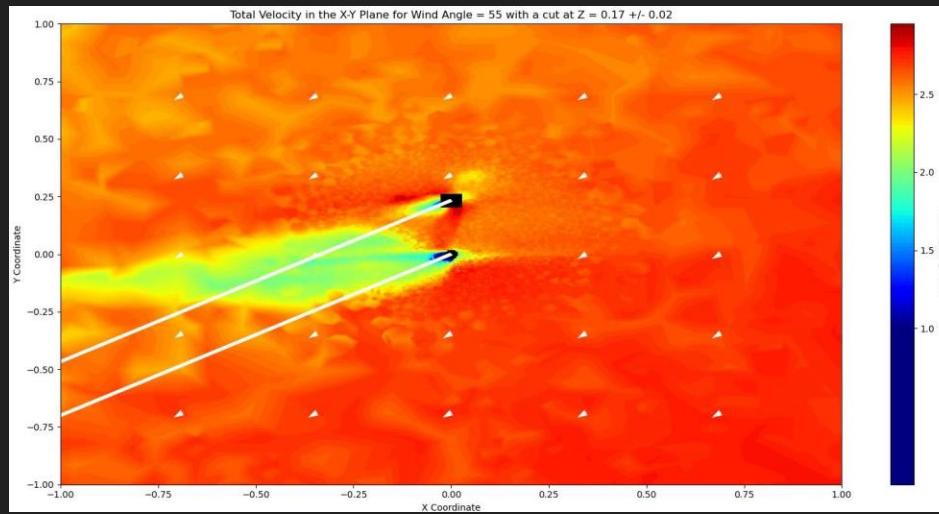


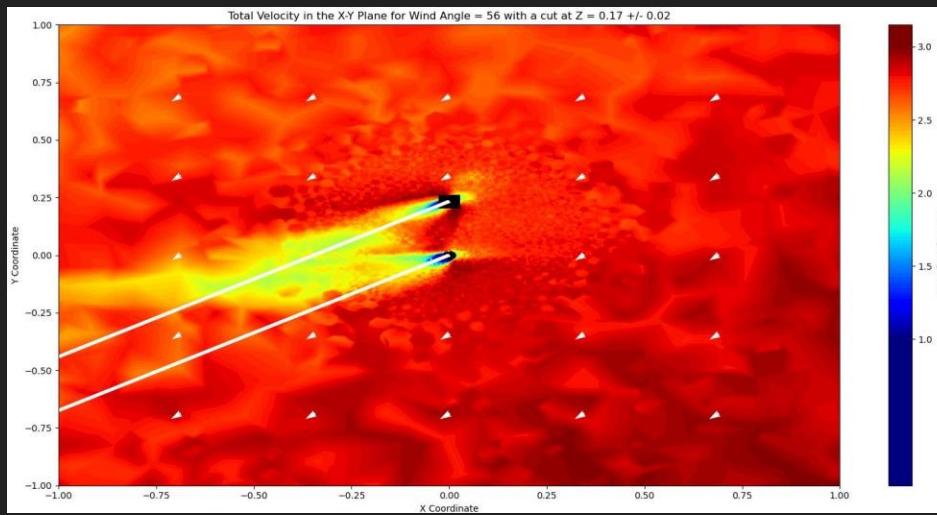


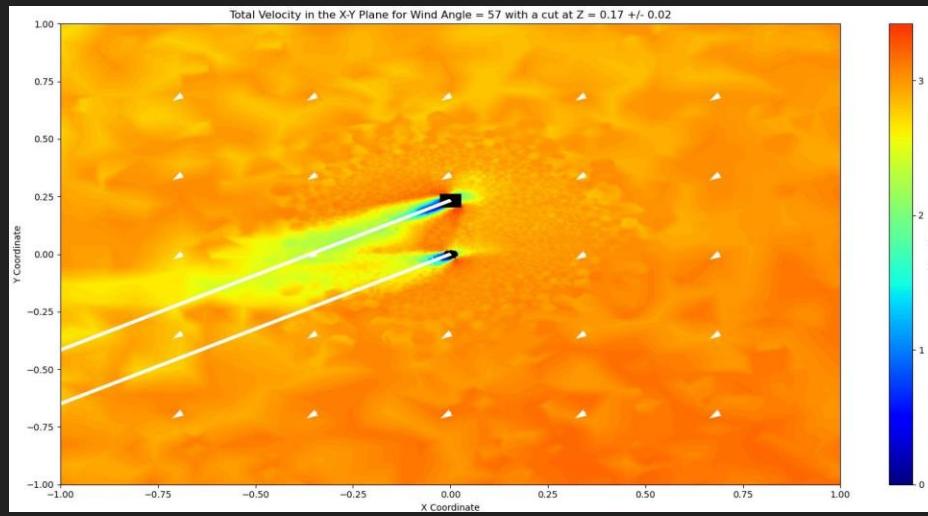


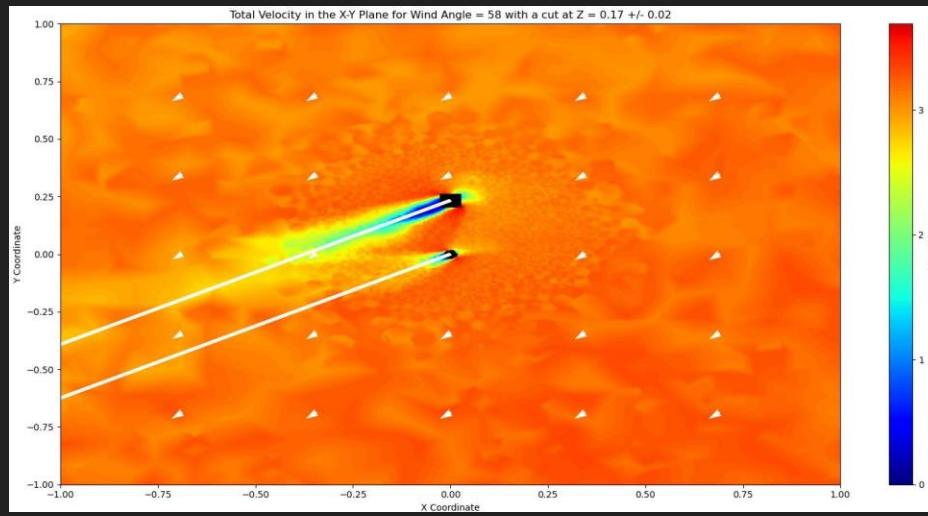


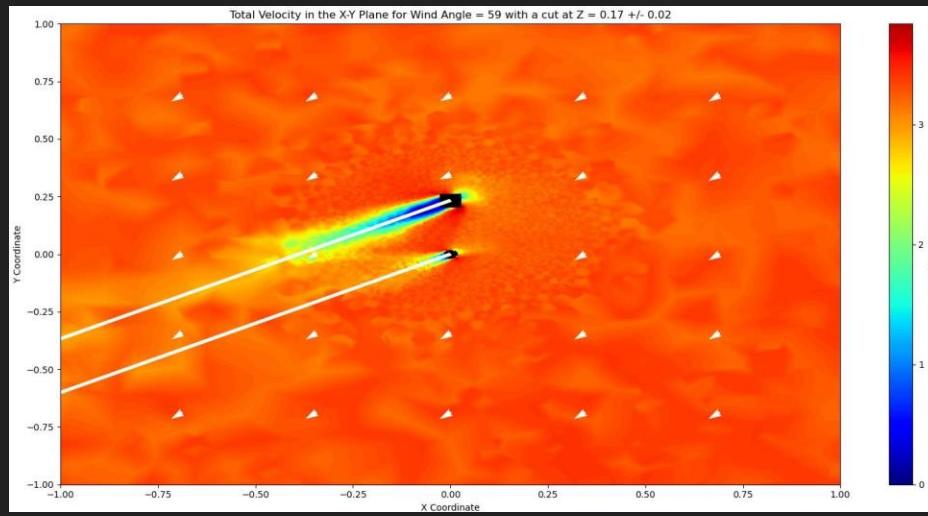


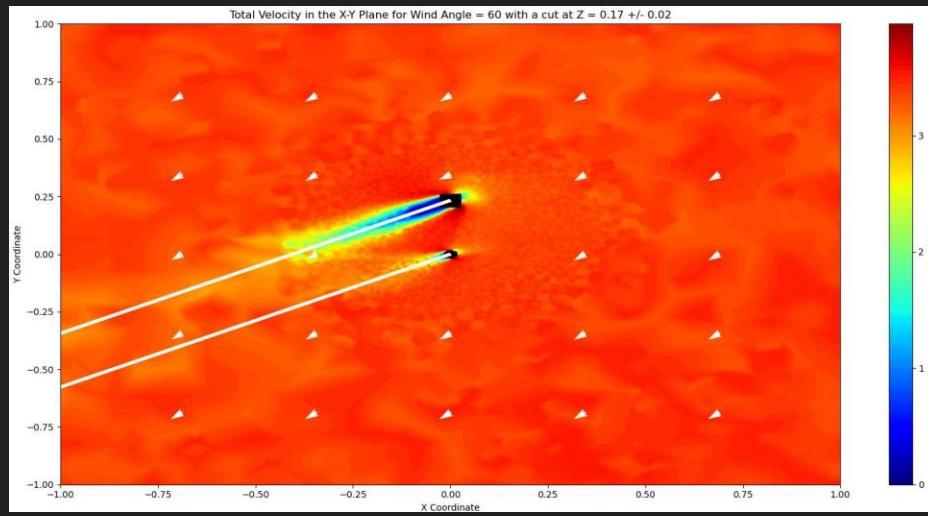


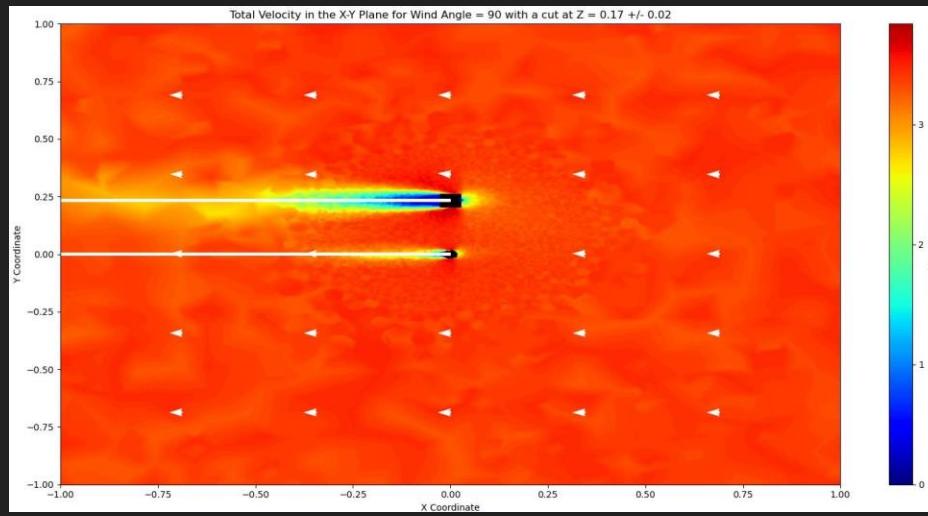


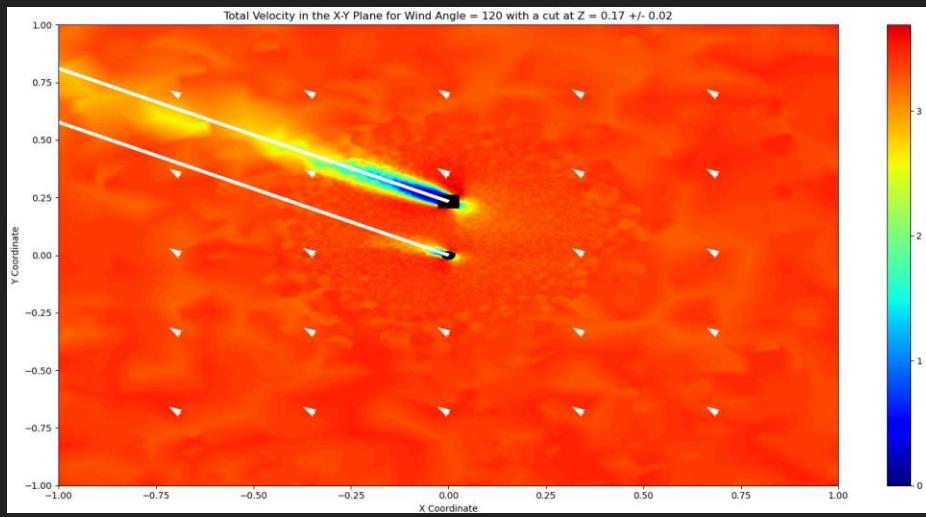


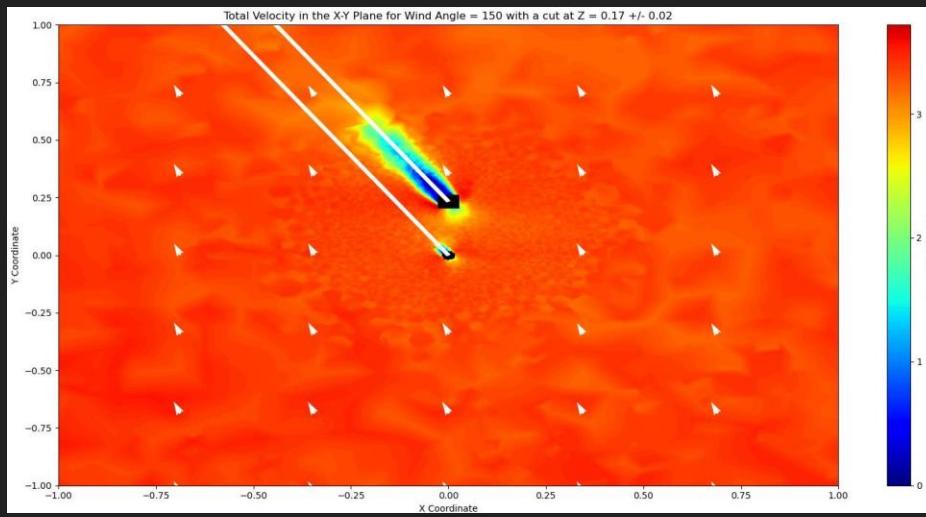


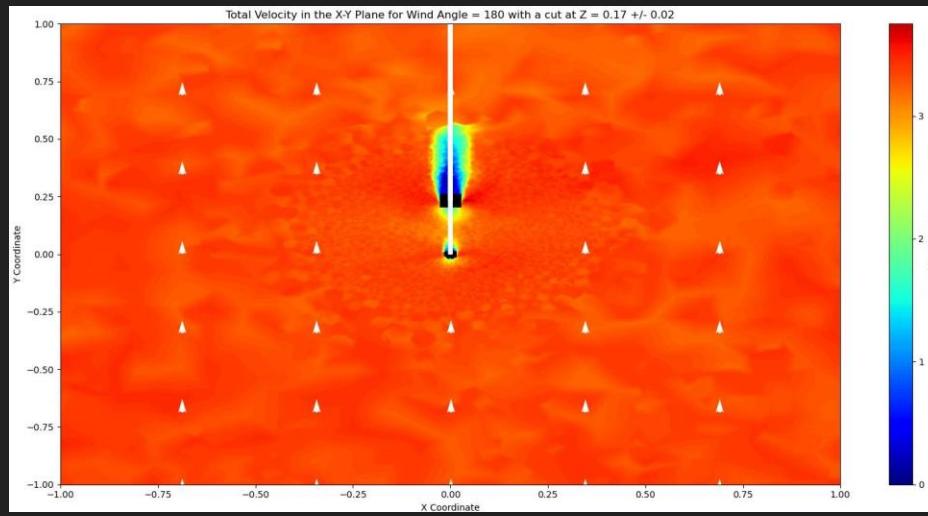


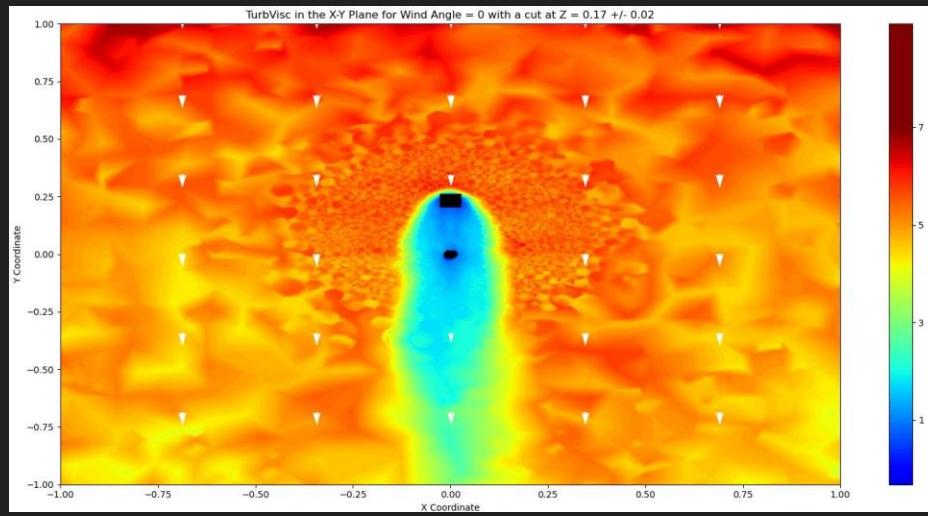


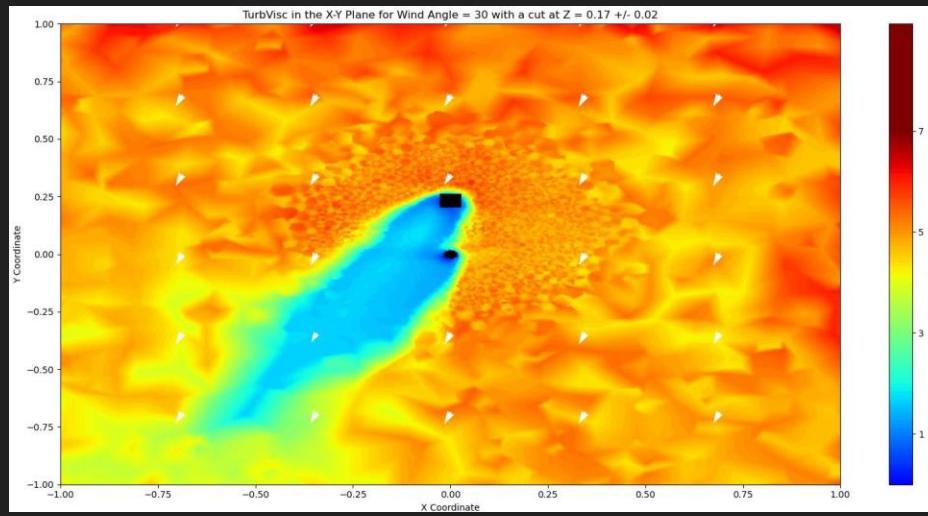


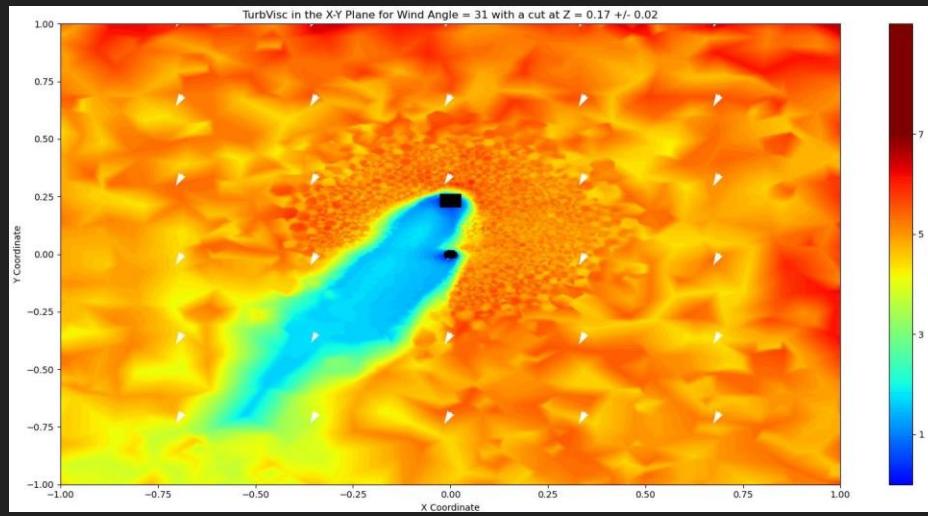


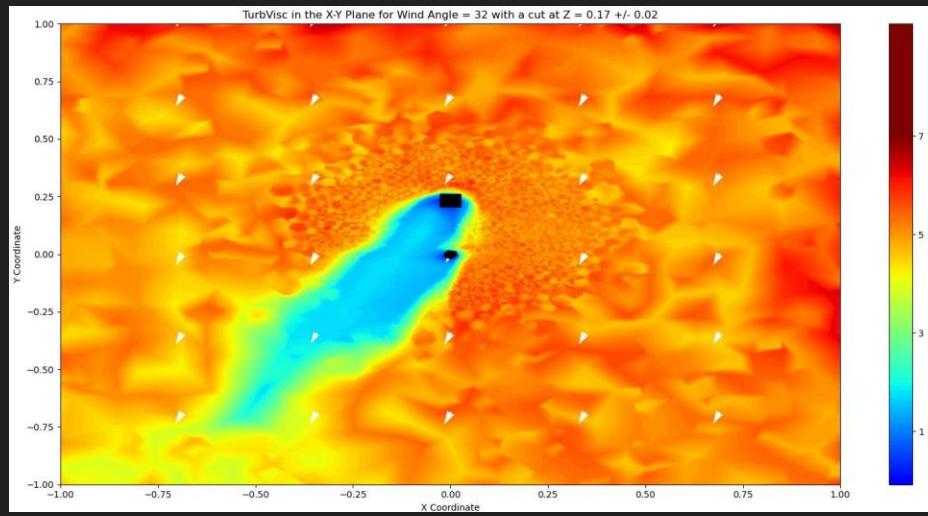


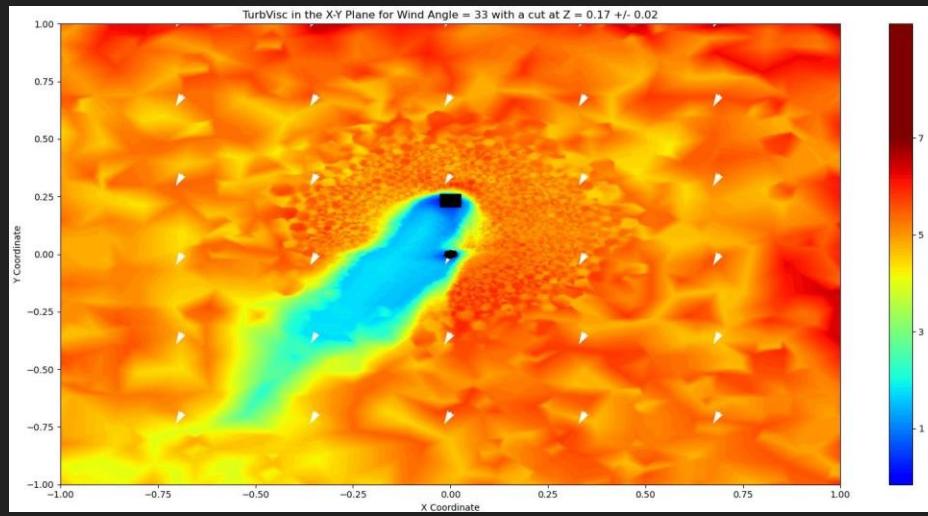


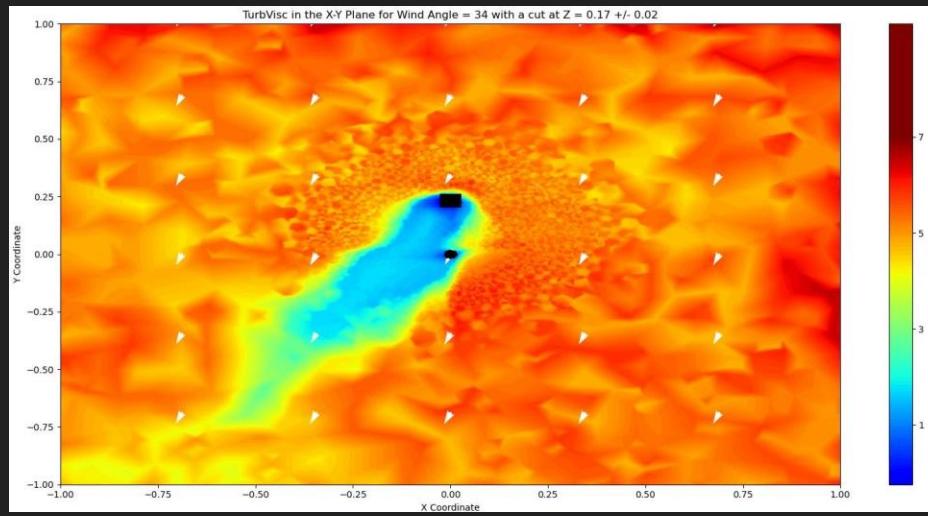


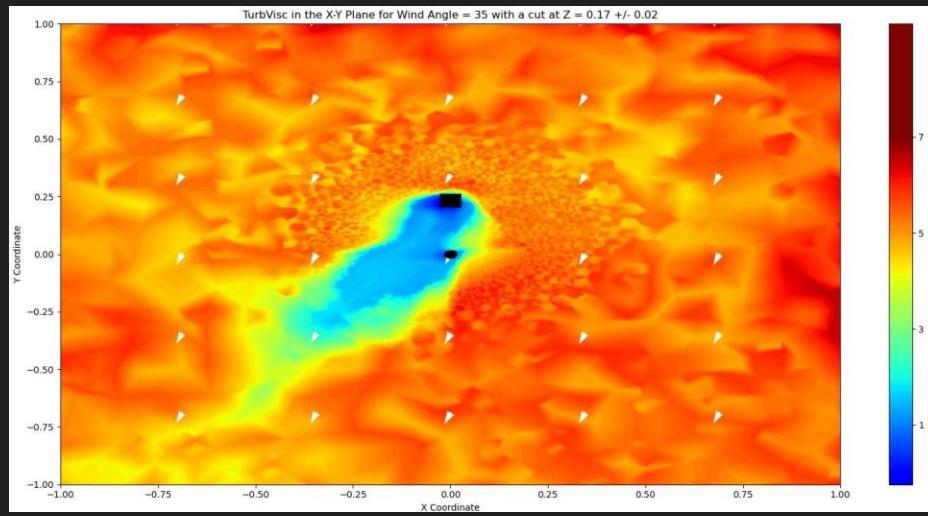


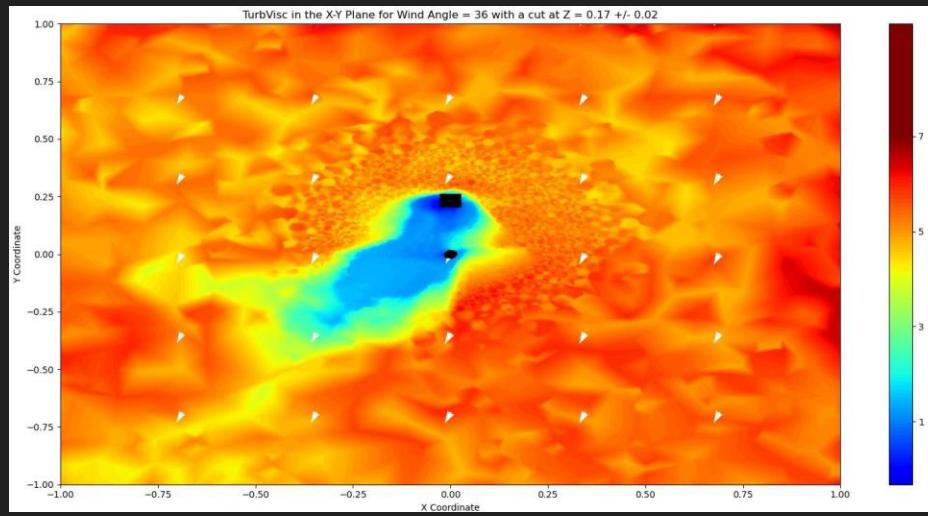


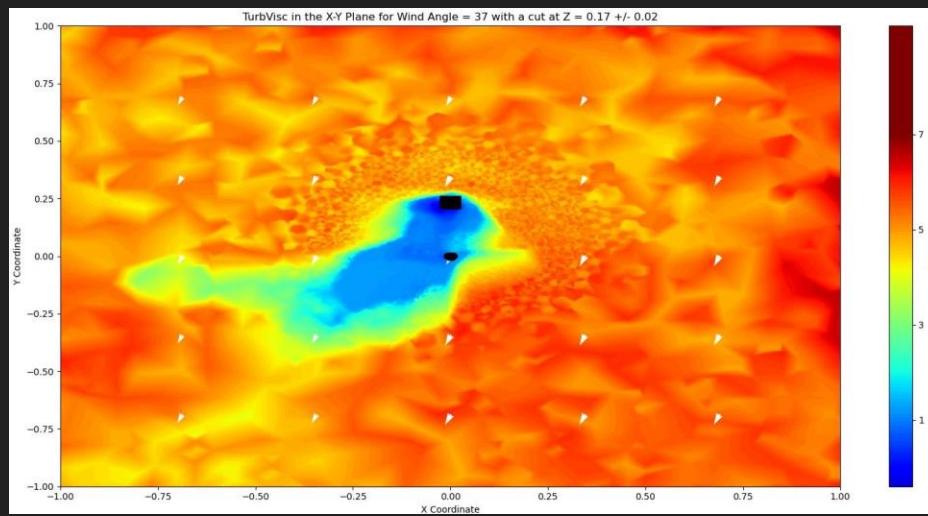


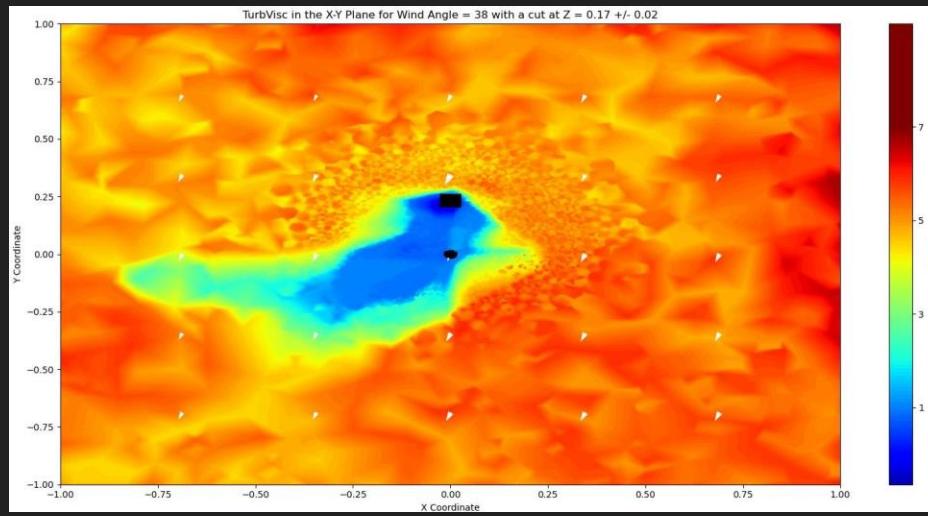


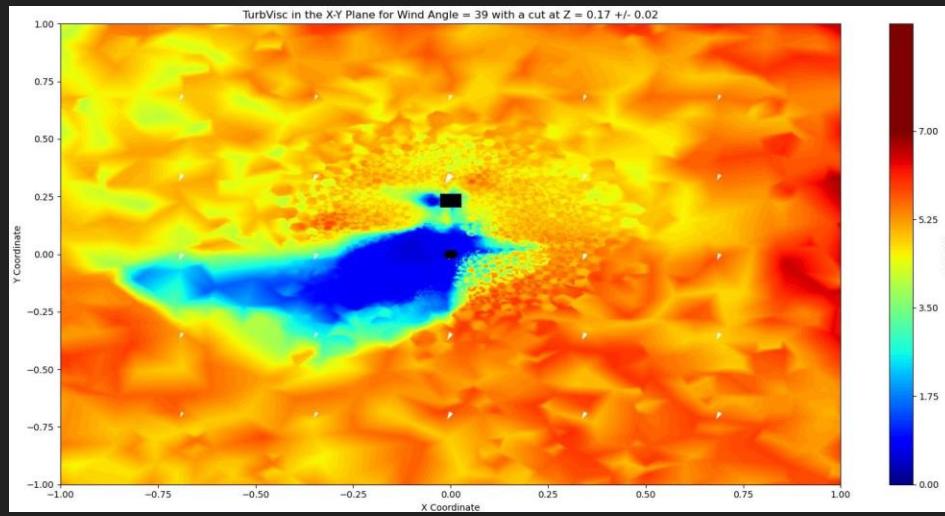


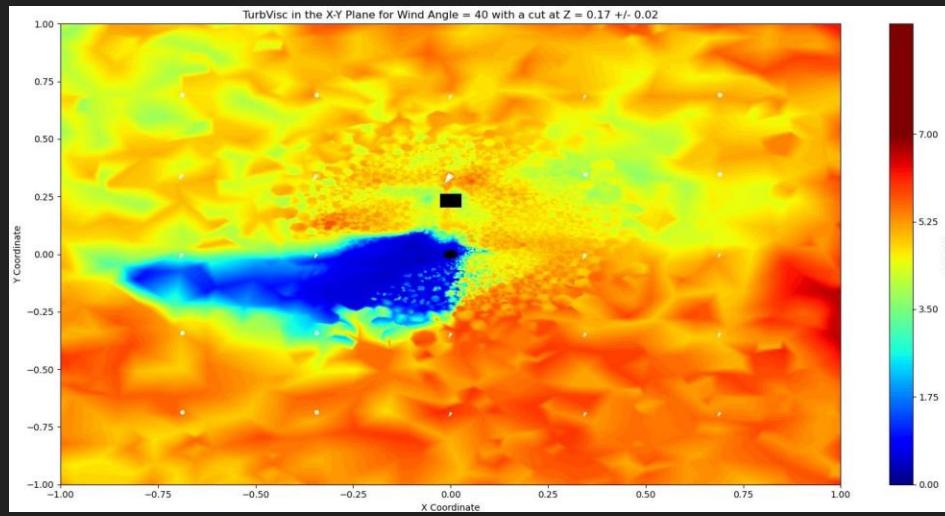


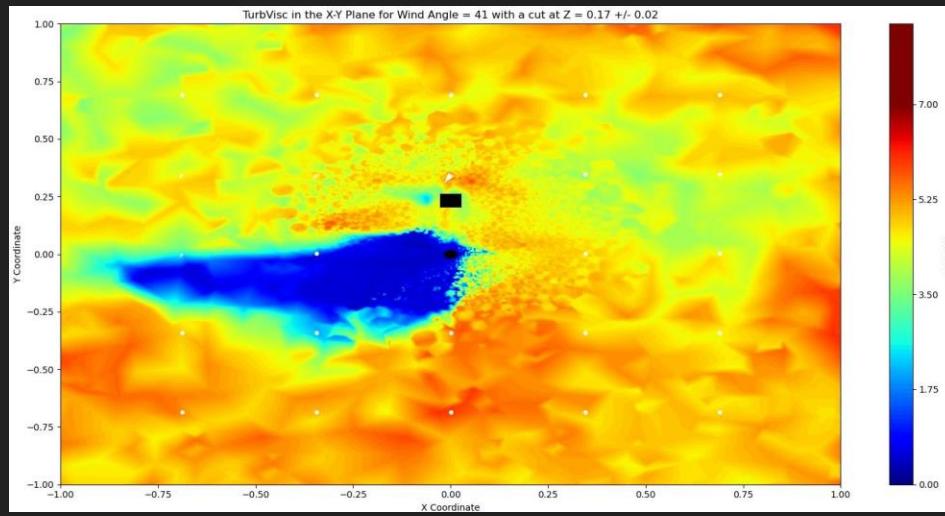


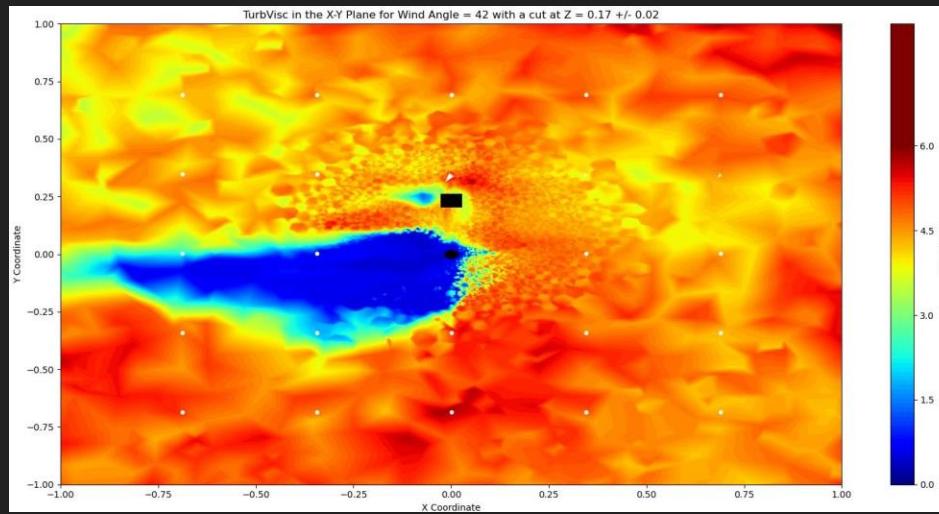


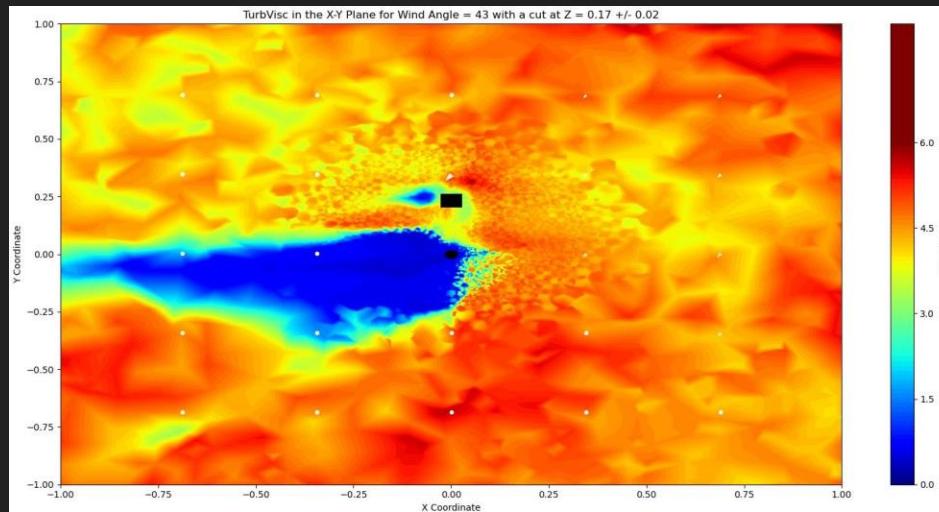


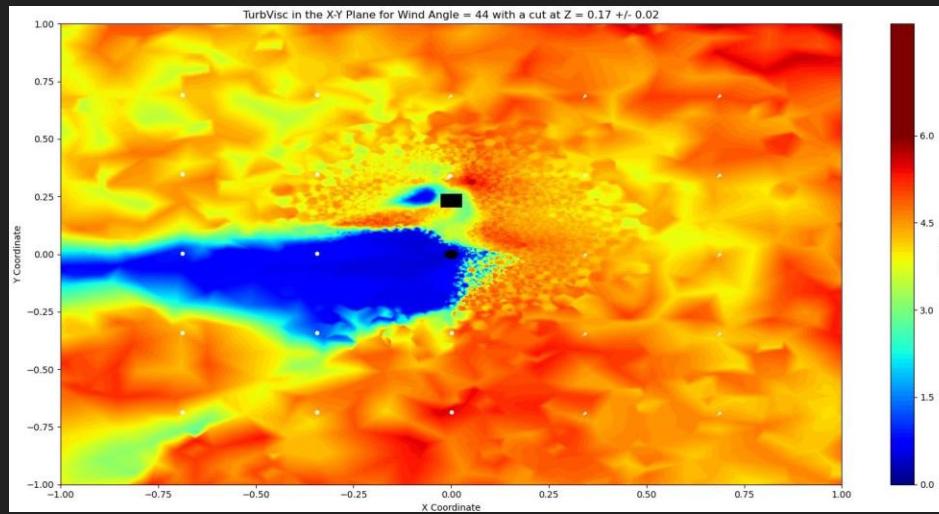


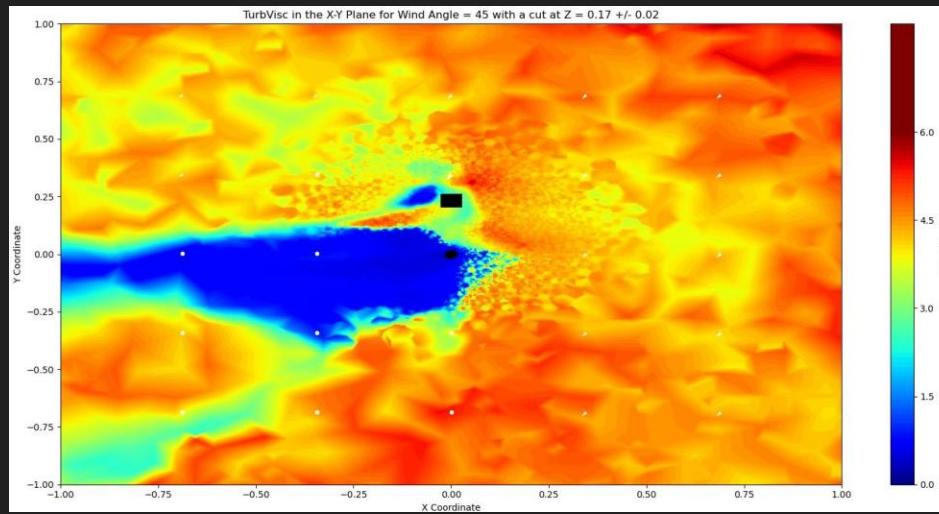


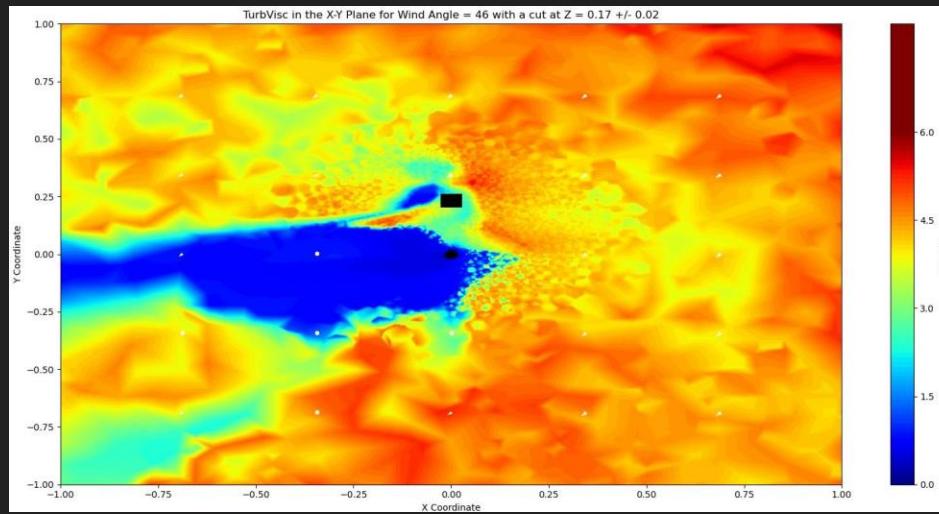


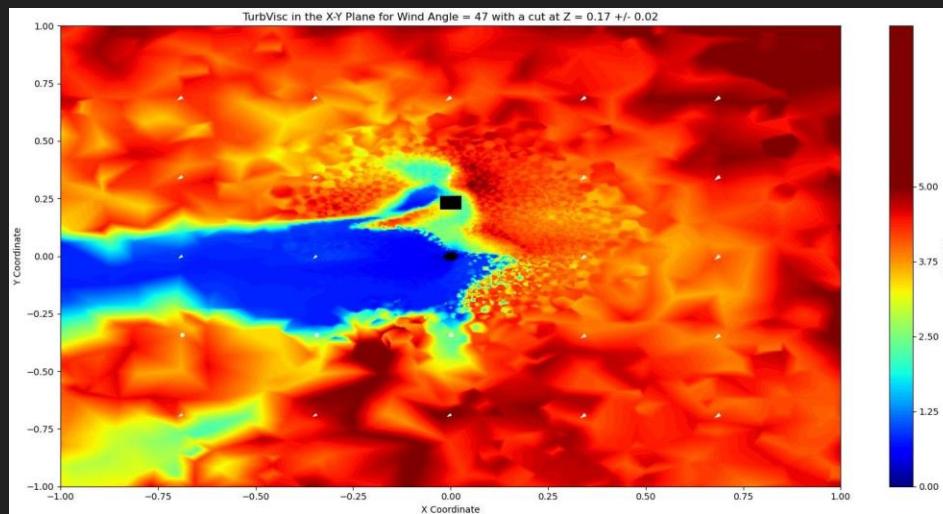


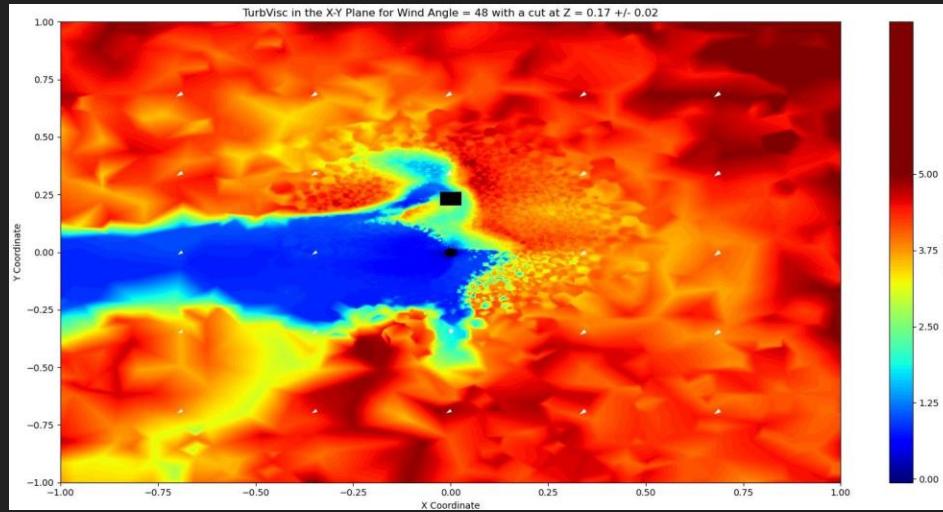


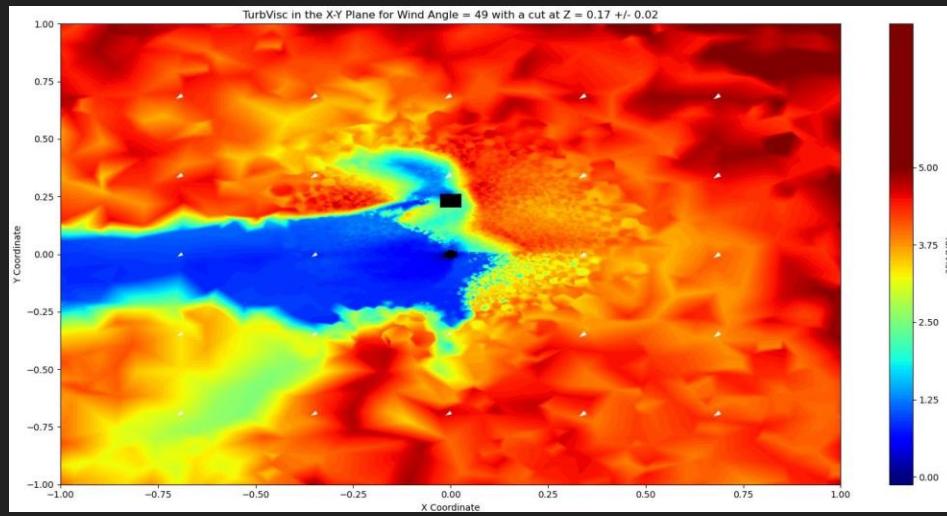


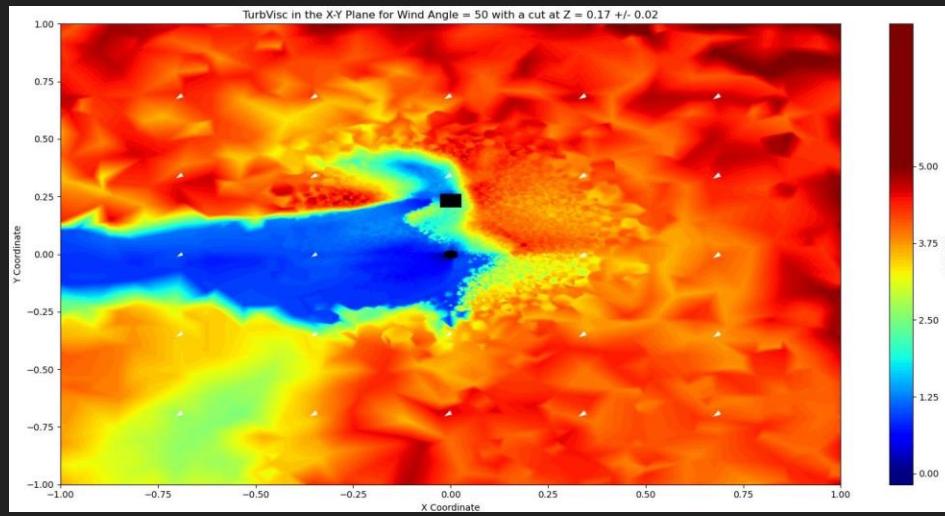


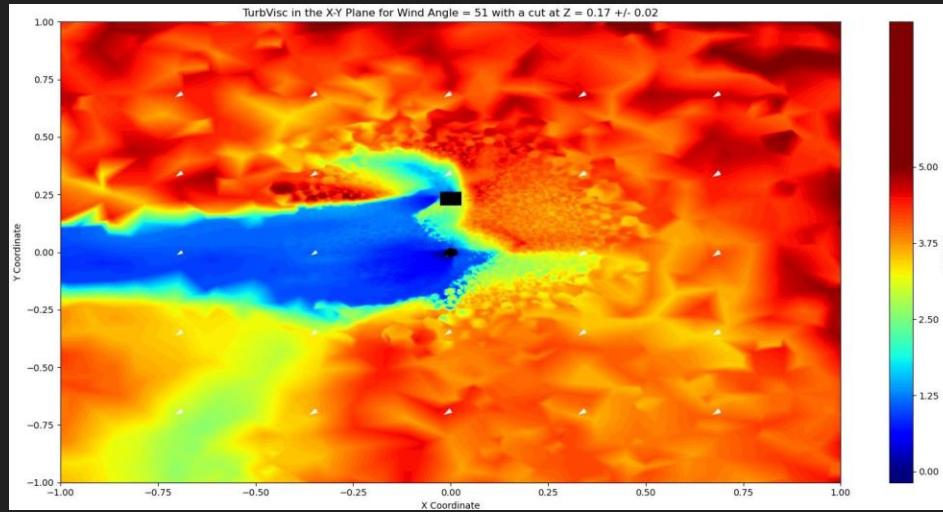


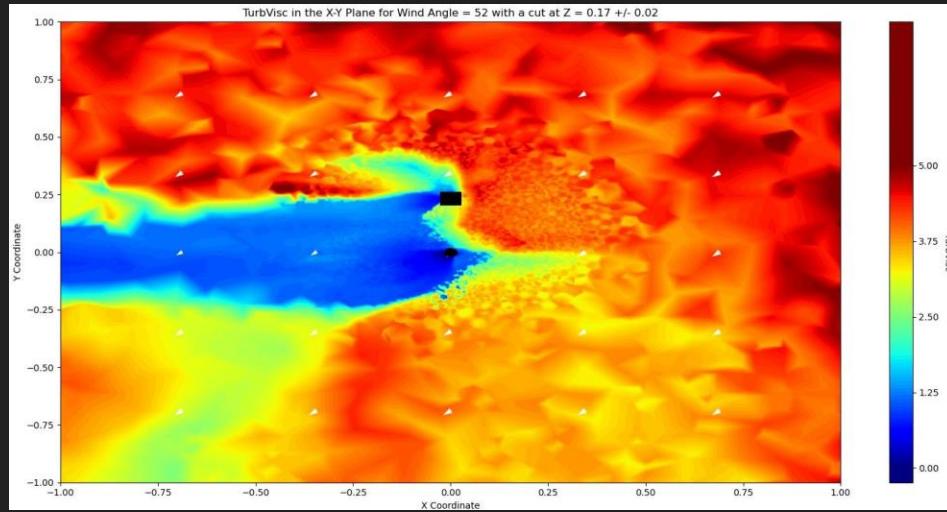


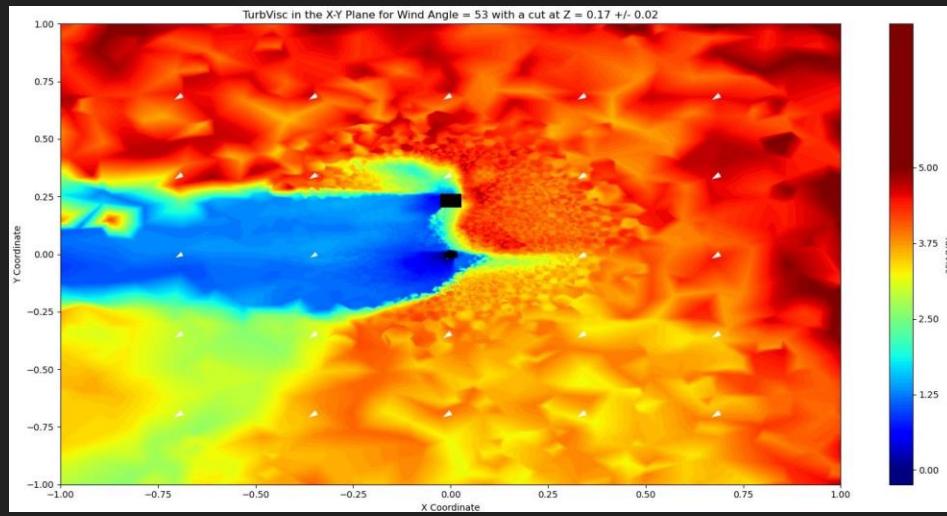


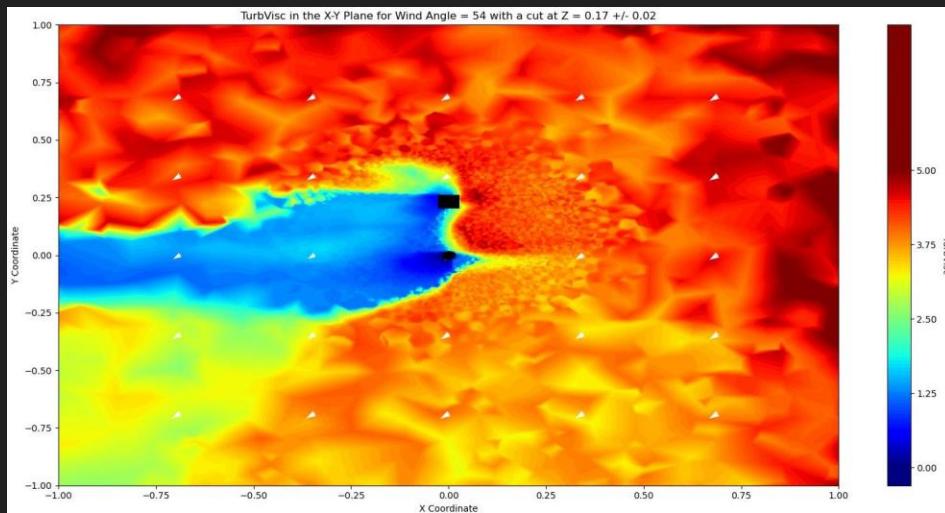


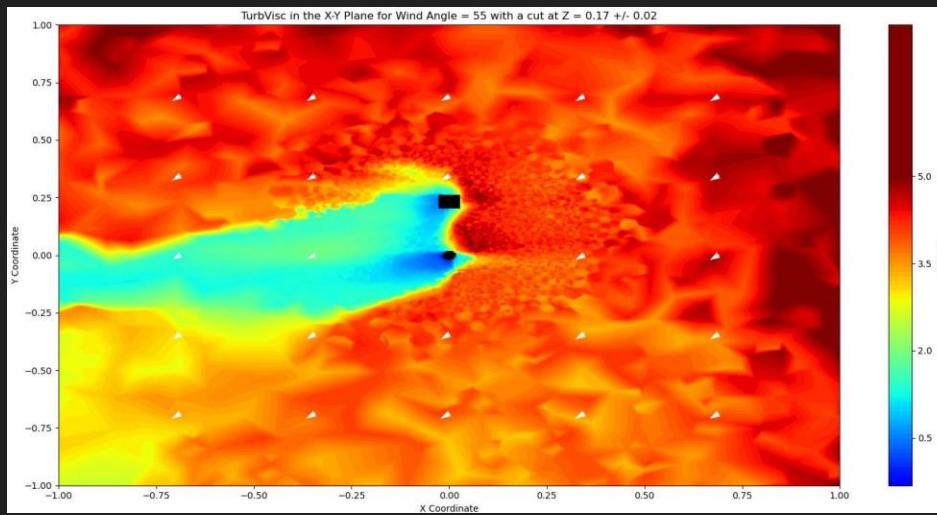


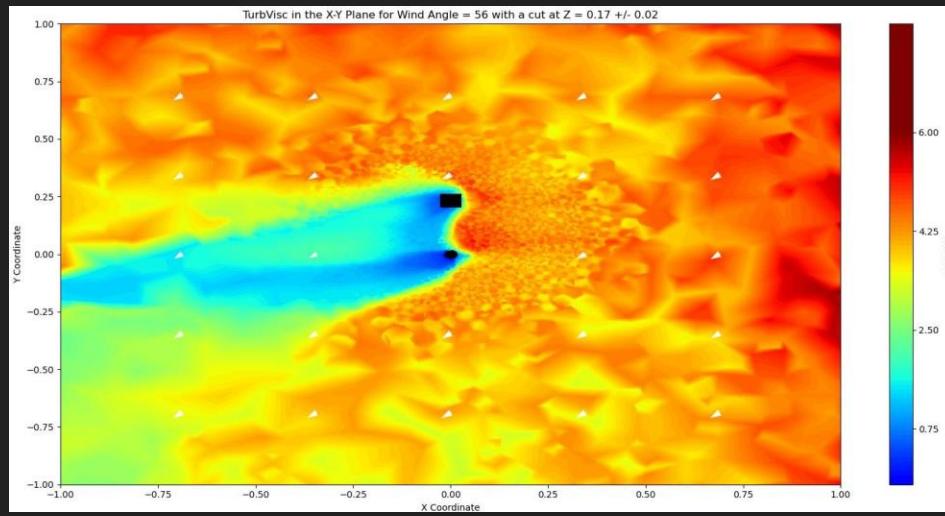


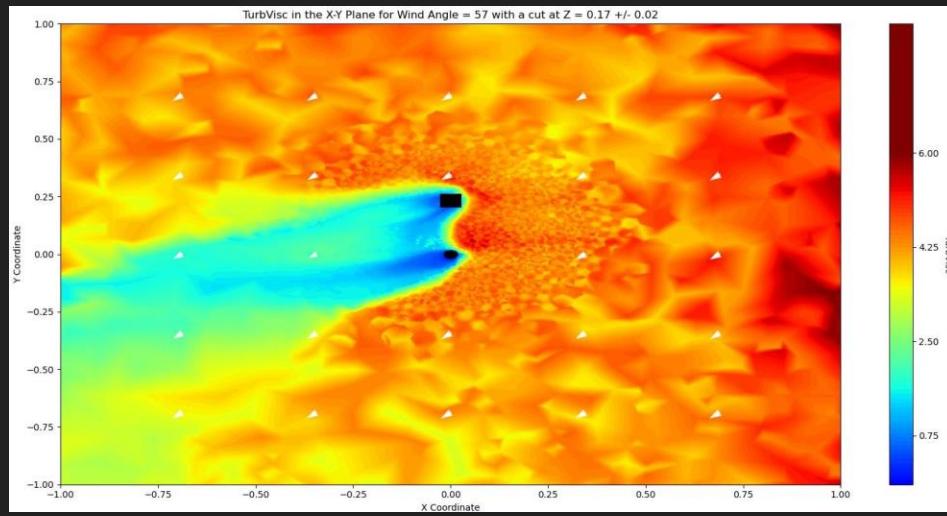


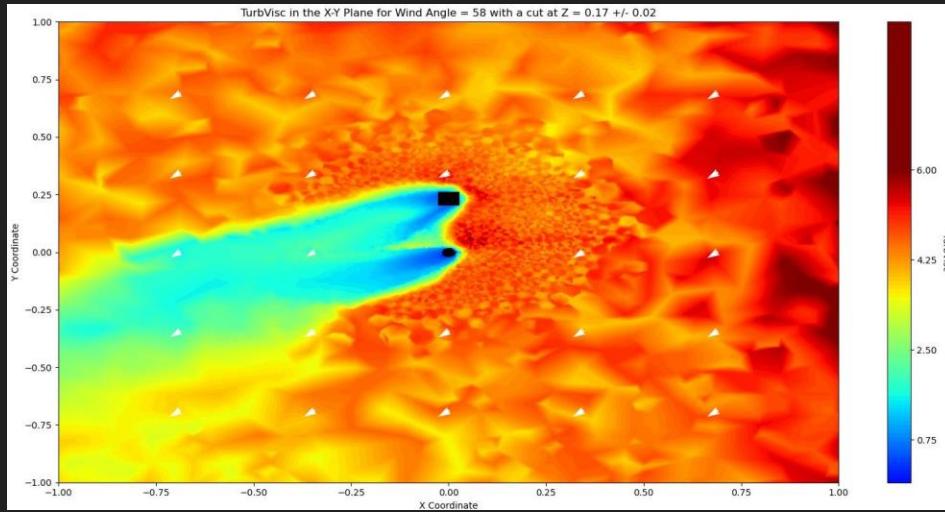


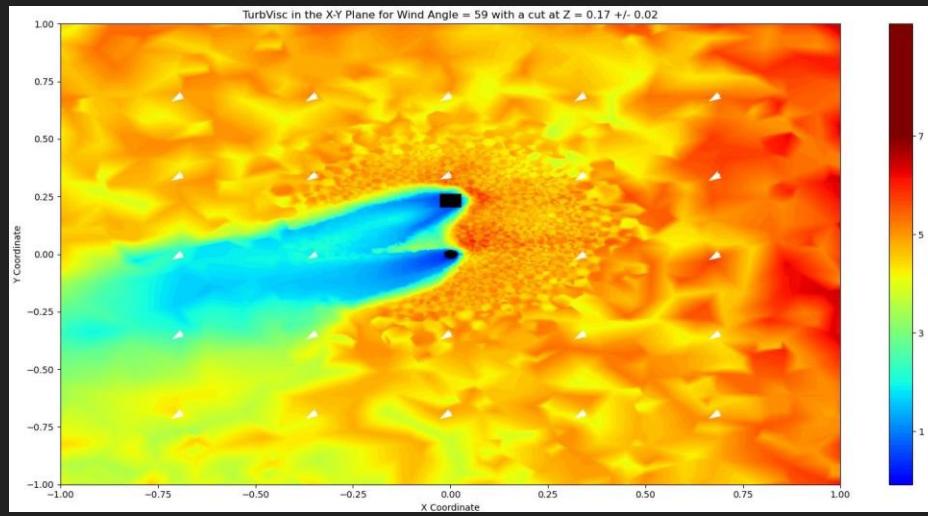


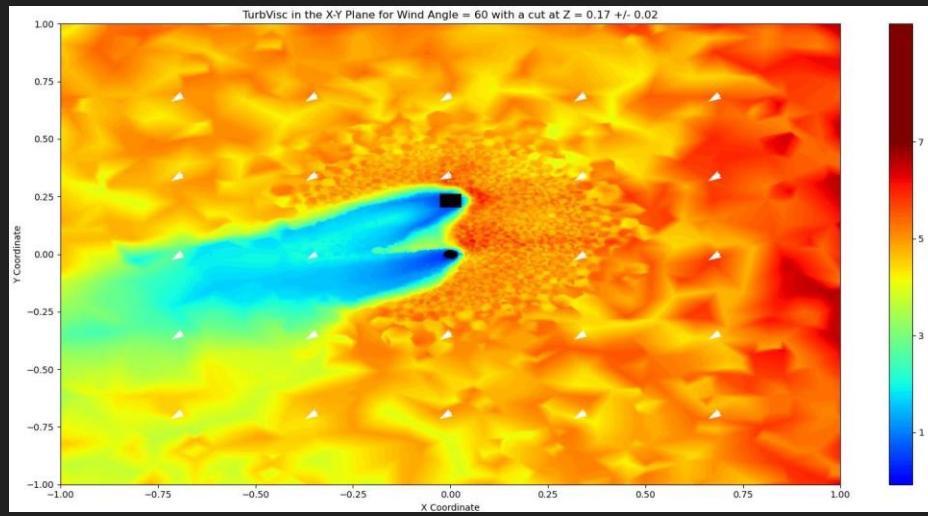


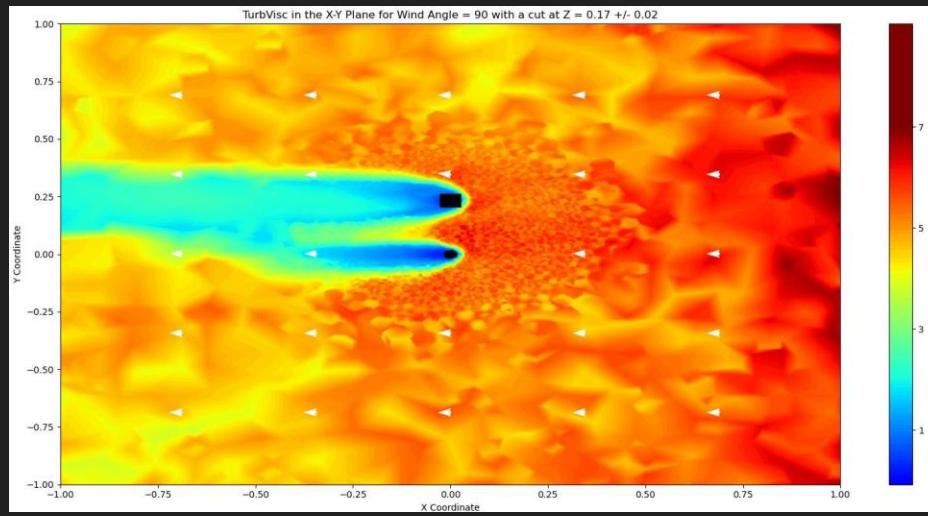


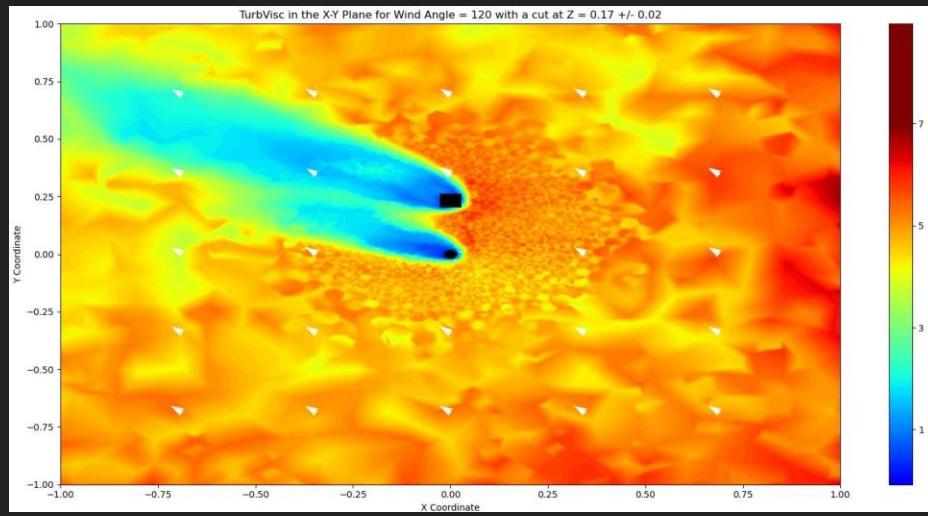


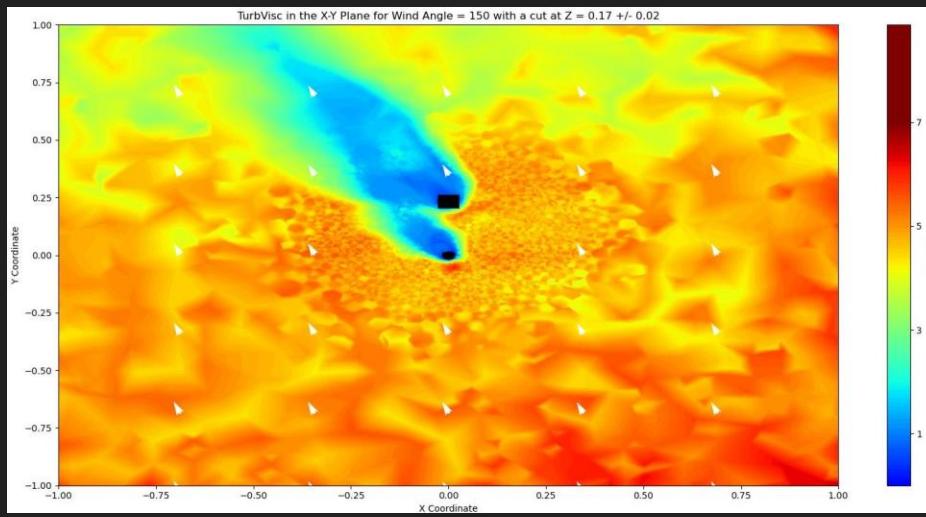


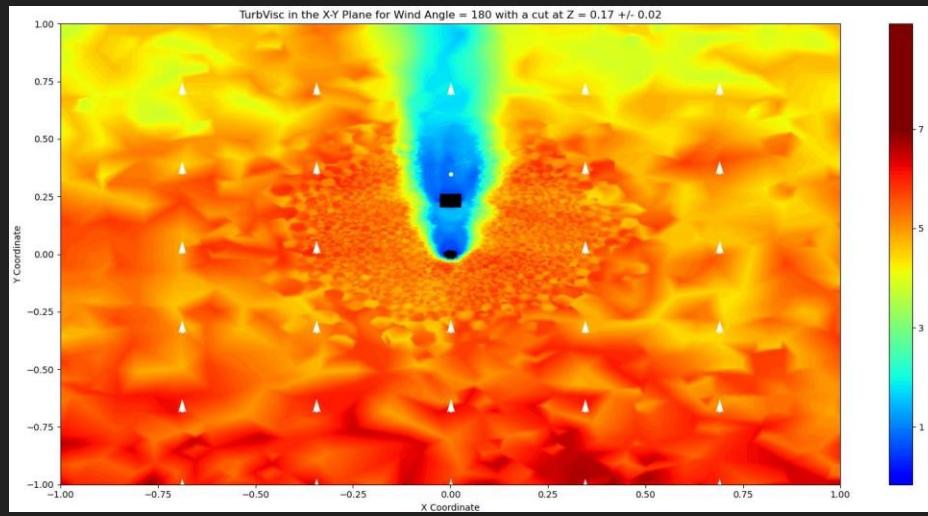












Progress so far - Data Loss + Cont Loss  
(with Boundary Conditions imposed)  
(Adam Optimizer)

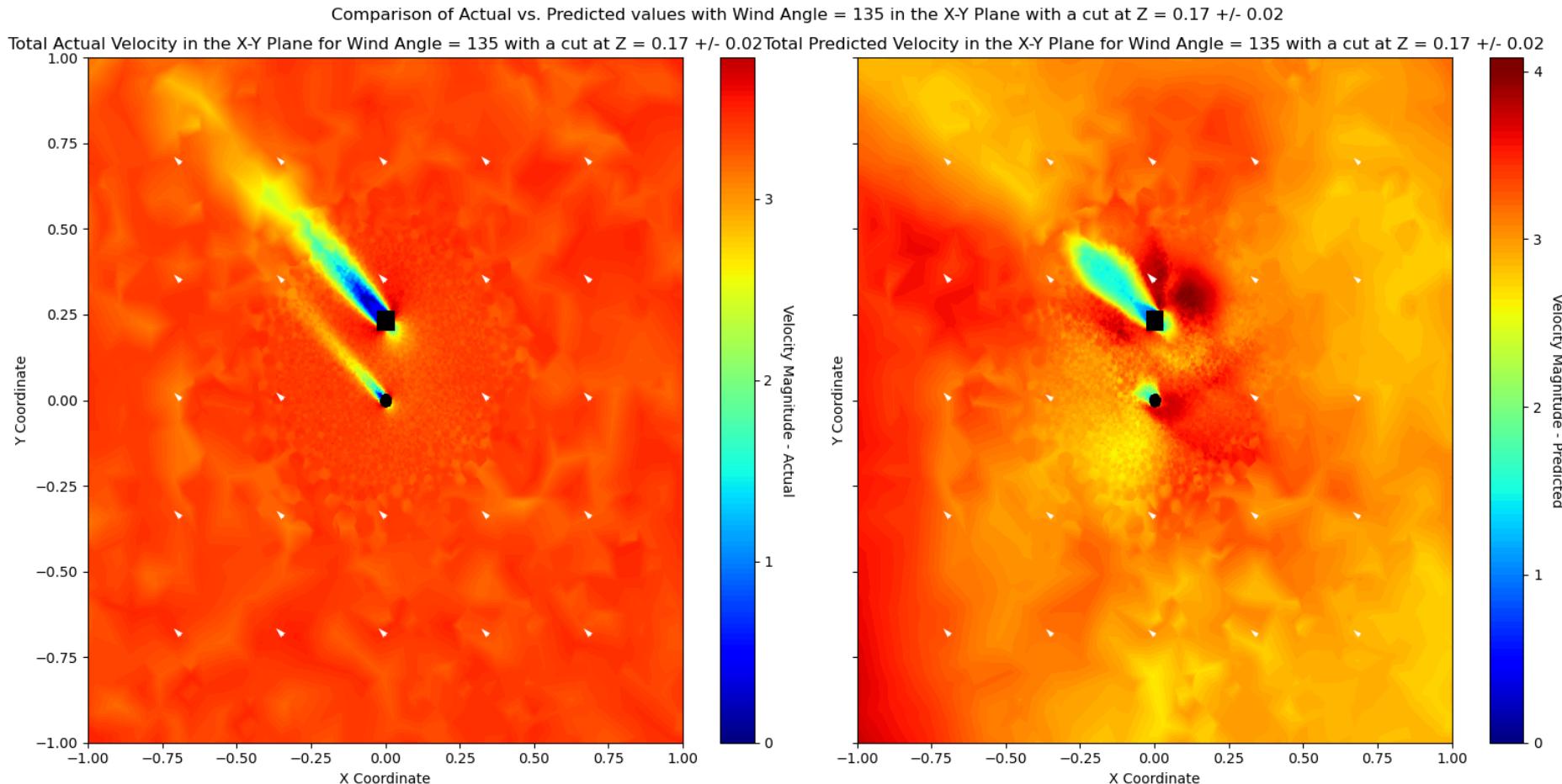
Threshold = 1E-5 (2800 Epochs, not completed), GPU Workstation

Scripts v3 – PREDICTING (135 DEG)

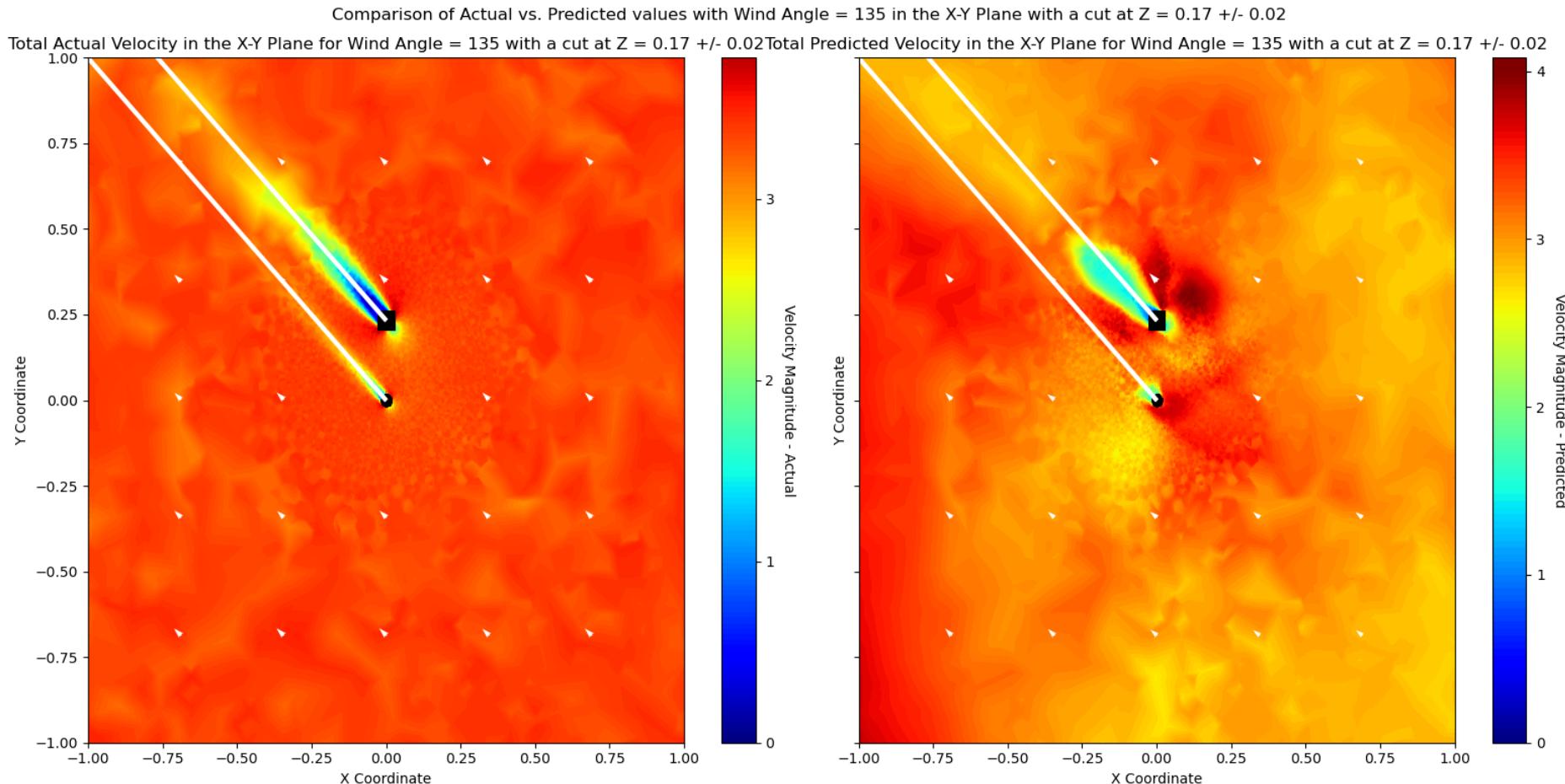
Progress so far - Data + Cont + Boundary Loss (Adam Optimizer)  
Threshold = 1E-5 (2800 Epochs, so far...), GPU Workstation  
Predicting Results – Metrics (Angle = 135)

Variable	MSE	RMSE	MAE	R2
Pressure	1.566548698	1.251618432	0.767839804	0.07324682
Velocity:0	0.152398519	0.390382529	0.272649504	0.850476404
Velocity:1	0.237096073	0.48692512	0.368208644	0.769594397
Velocity:2	0.01174556	0.108376934	0.056323591	0.640962943
TurbVisc	2.331652589	1.52697498	1.142492203	0.983009331

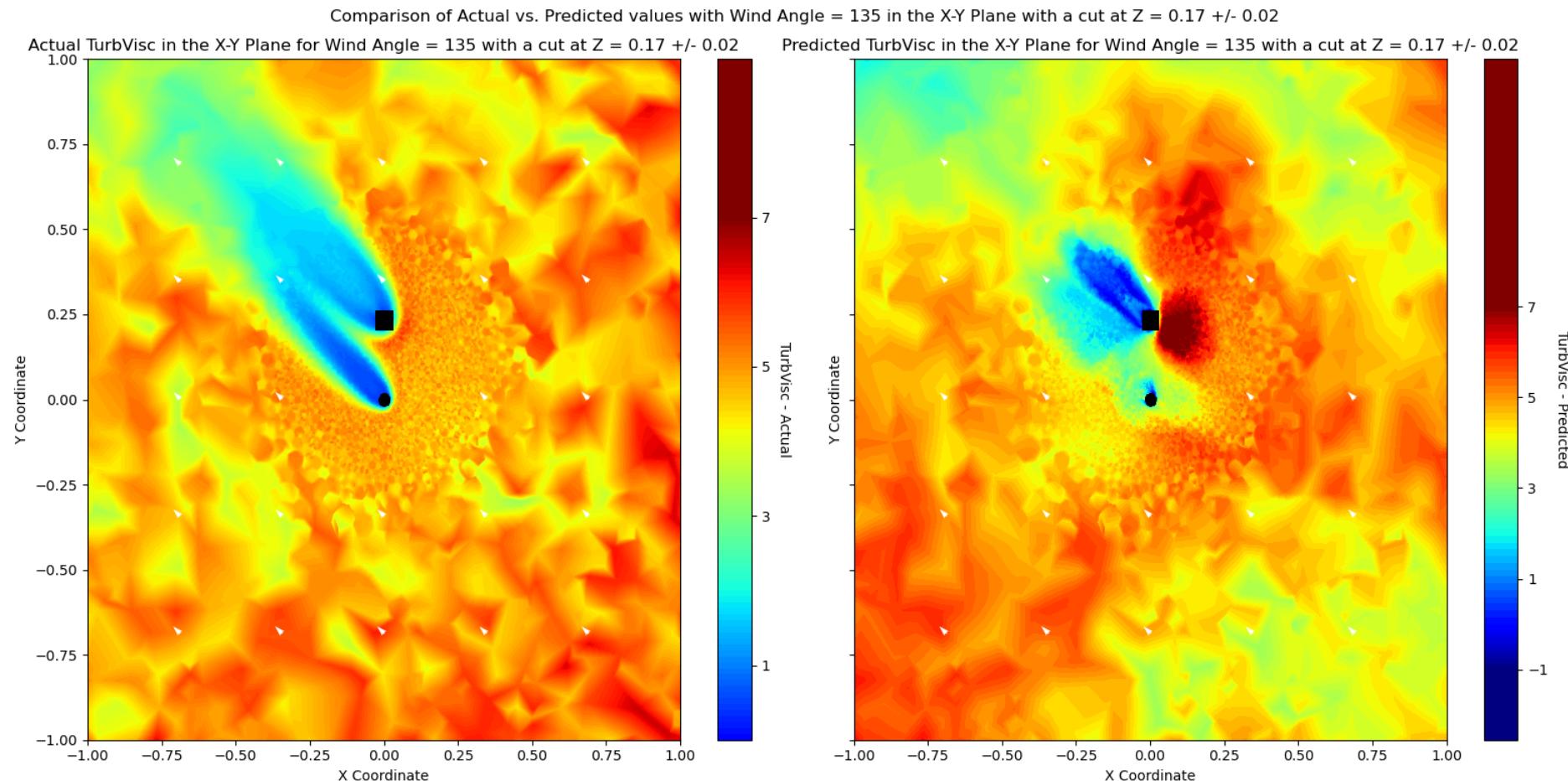
Progress so far - Data + Cont + Boundary Loss (Adam Optimizer), Threshold = 1E-5 (2800 Epochs, so far...), GPU Workstation  
Predicting Results - X-Y Total Velocity Plot (Angle = 135)



Progress so far - Data + Cont + Boundary Loss (Adam Optimizer), Threshold = 1E-5 (2800 Epochs, so far...), GPU Workstation  
Predicting Results - X-Y Total Velocity Plot (Angle = 135)

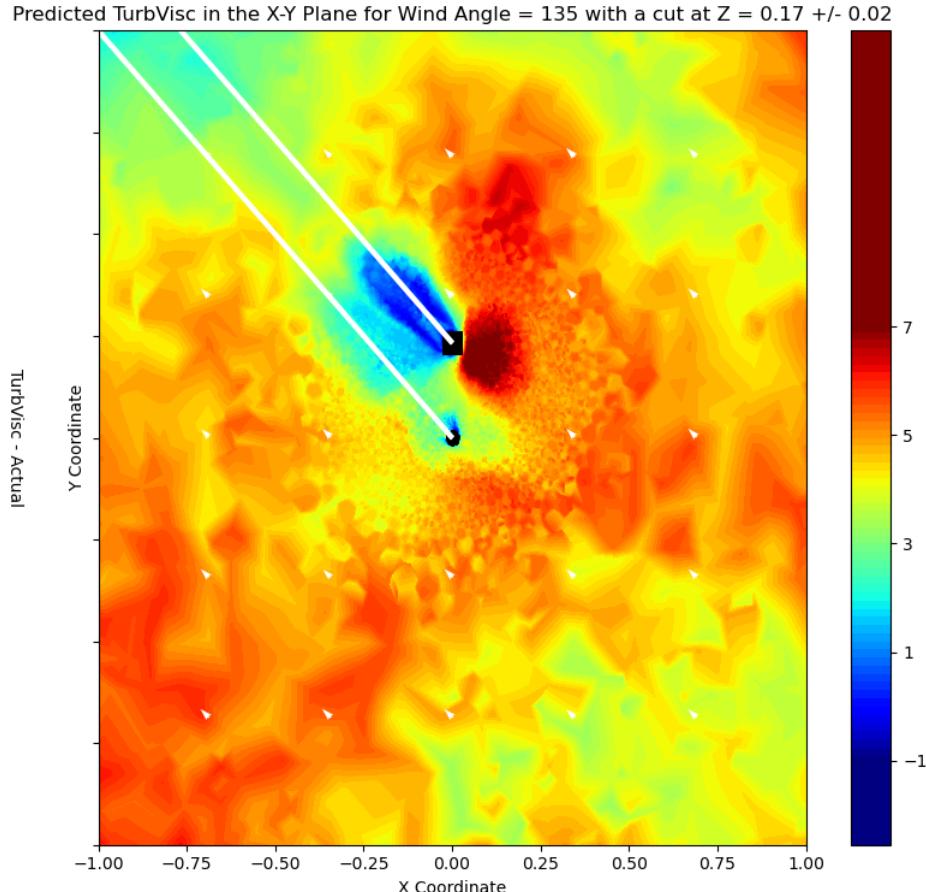
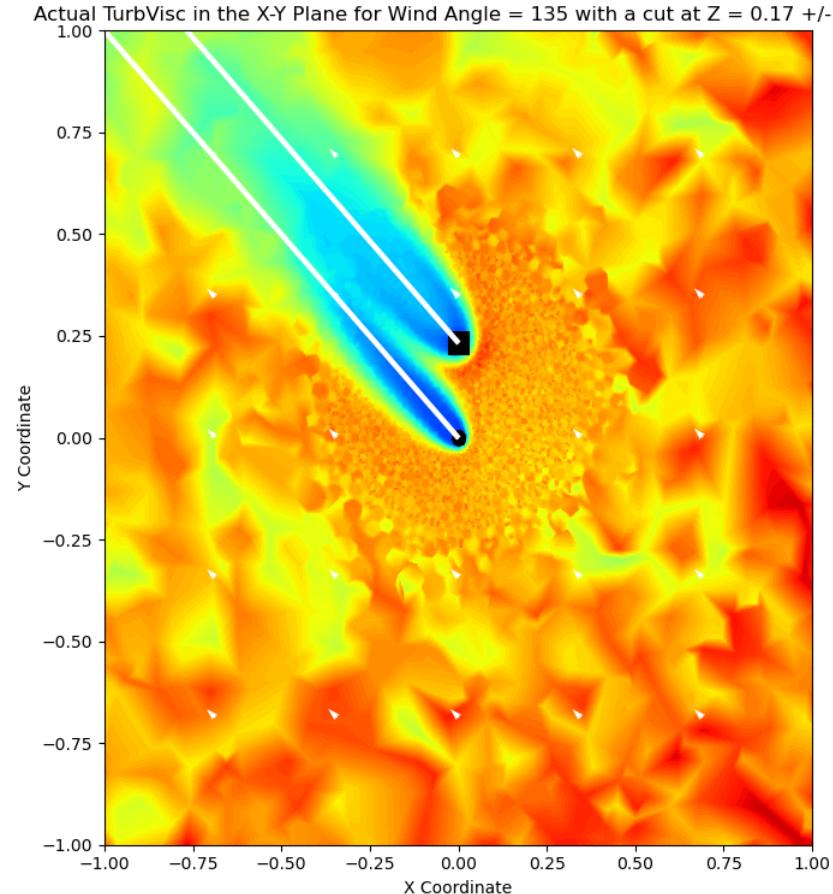


Progress so far - Data + Cont + Boundary Loss (Adam Optimizer), Threshold = 1E-5 (2800 Epochs, so far...), GPU Workstation  
Predicting Results - X-Y TurbVisc Plot (Angle = 135)



Progress so far - Data + Cont + Boundary Loss (Adam Optimizer), Threshold = 1E-5 (2800 Epochs, so far...), GPU Workstation  
Predicting Results - X-Y TurbVisc Plot (Angle = 135)

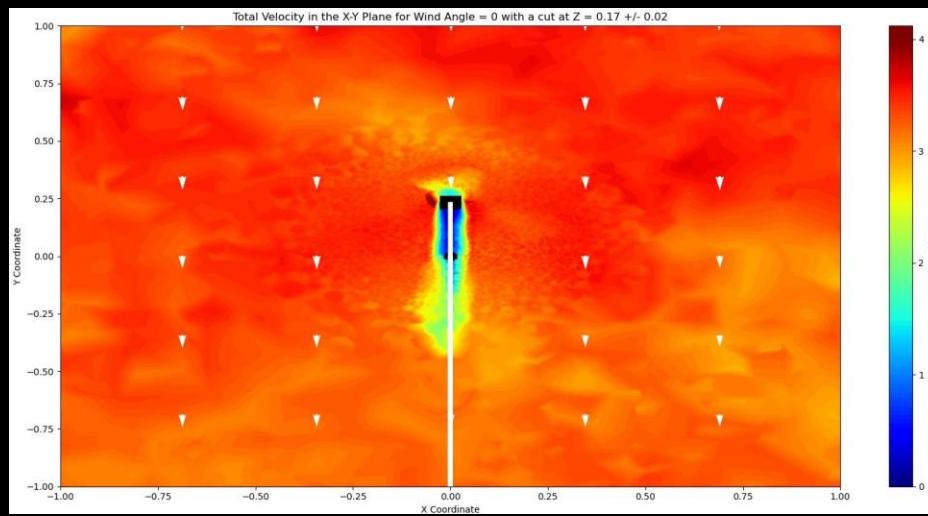
Comparison of Actual vs. Predicted values with Wind Angle = 135 in the X-Y Plane with a cut at Z = 0.17 +/- 0.02

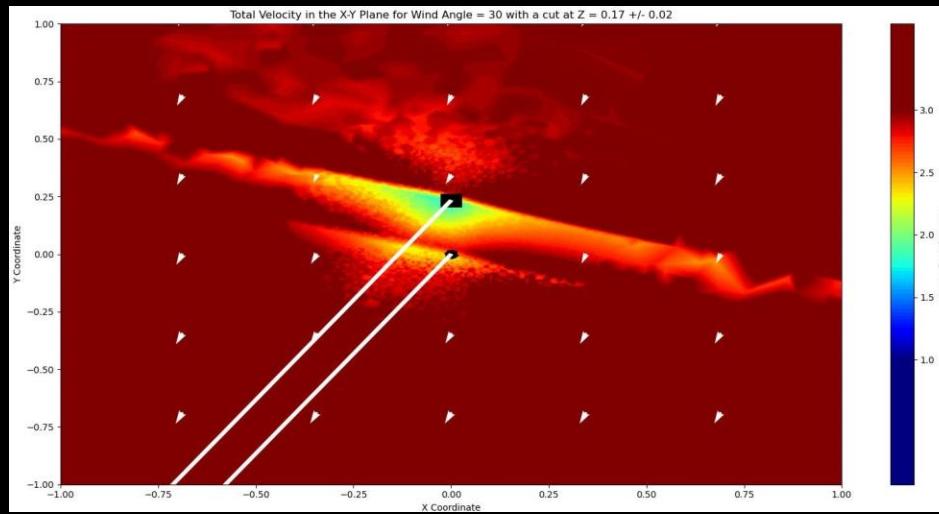


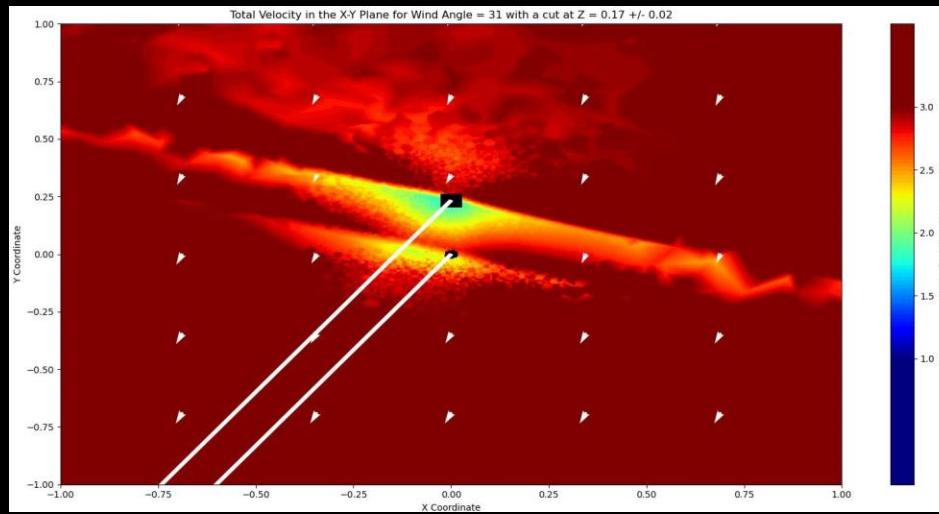
Progress so far - Data Loss + Cont Loss  
(with Boundary Conditions imposed)  
(Adam Optimizer)

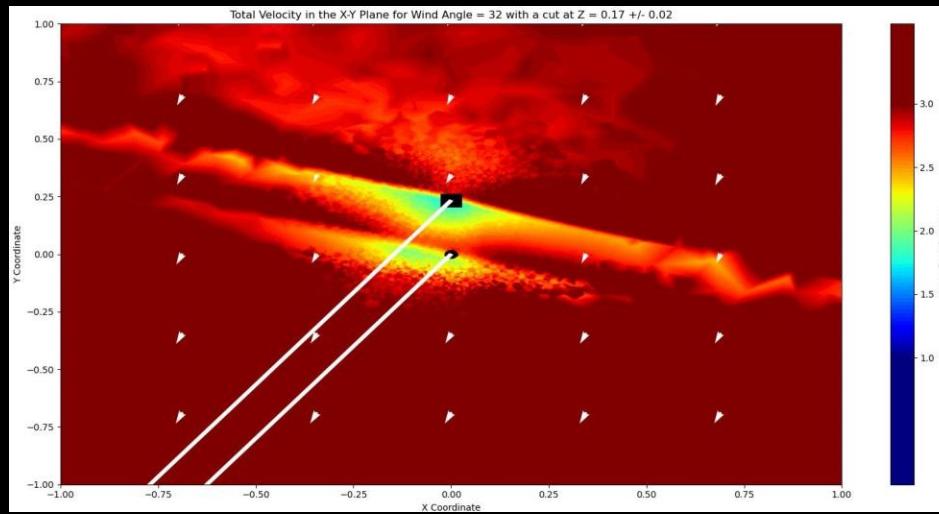
Threshold = 1E-5 (2800 Epochs, not completed), GPU Workstation

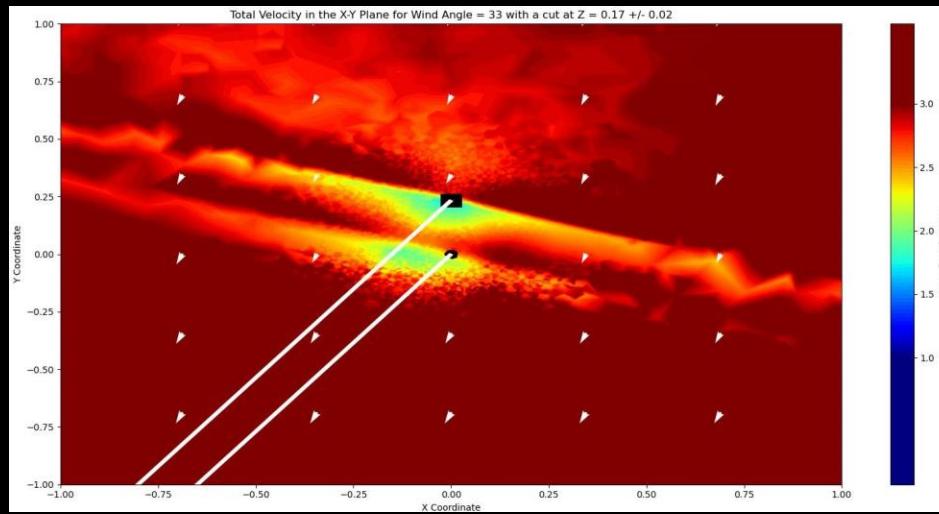
Scripts v3 – Plotting Any Angle

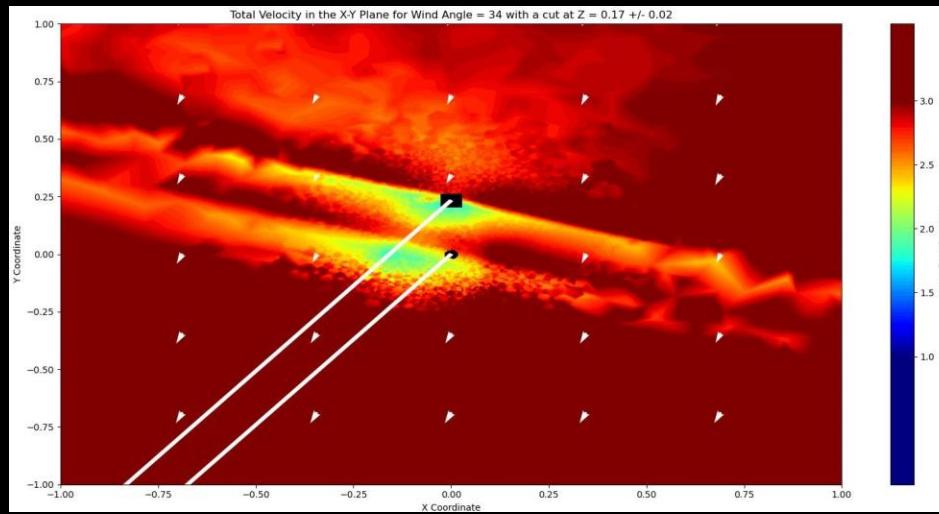


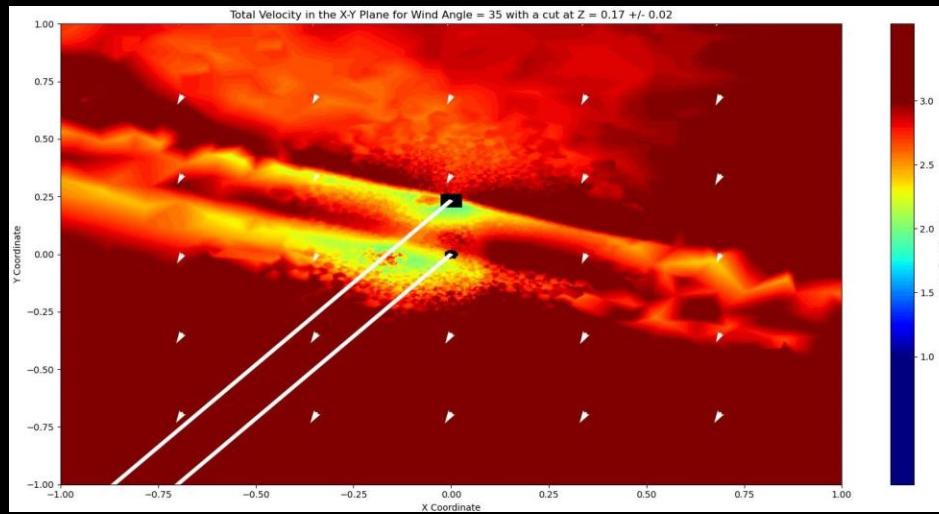


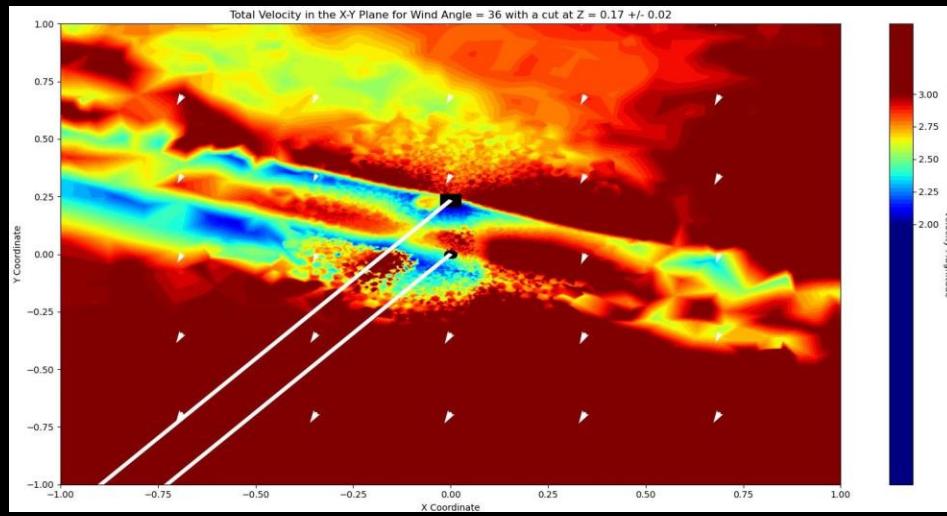


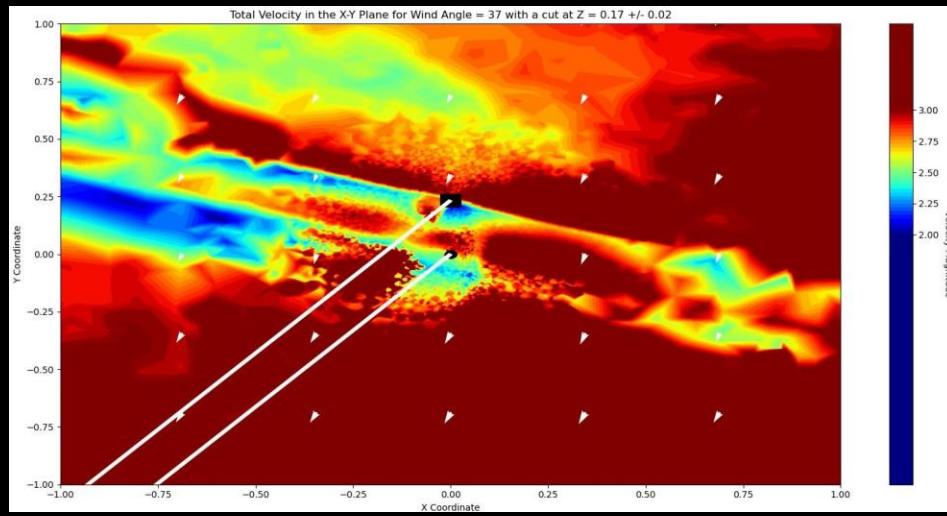


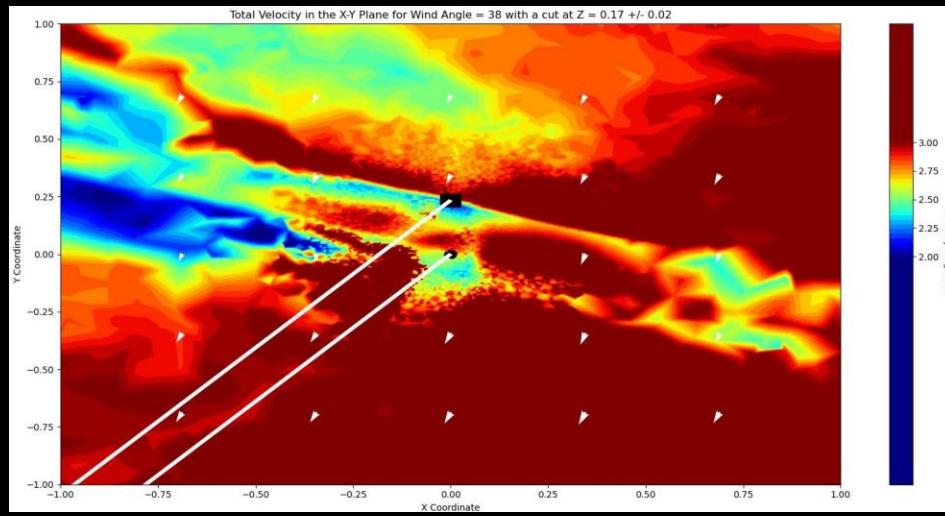


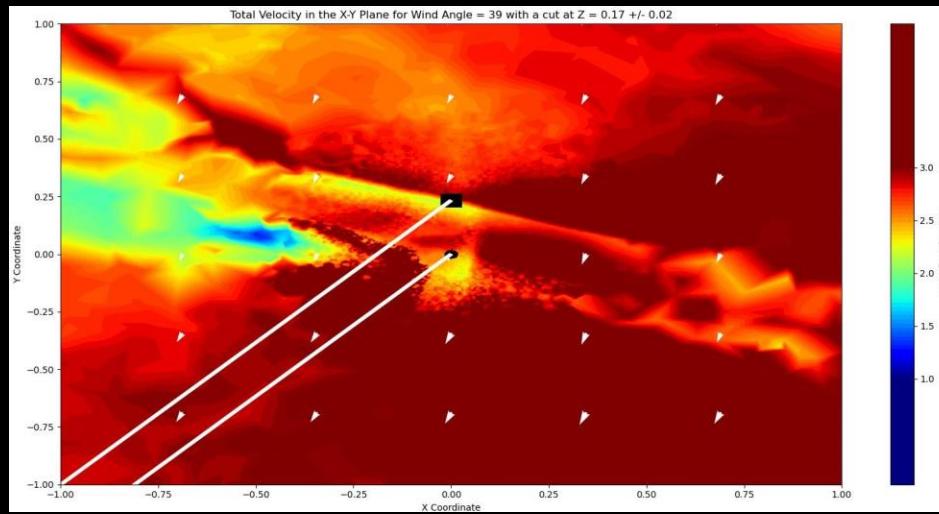


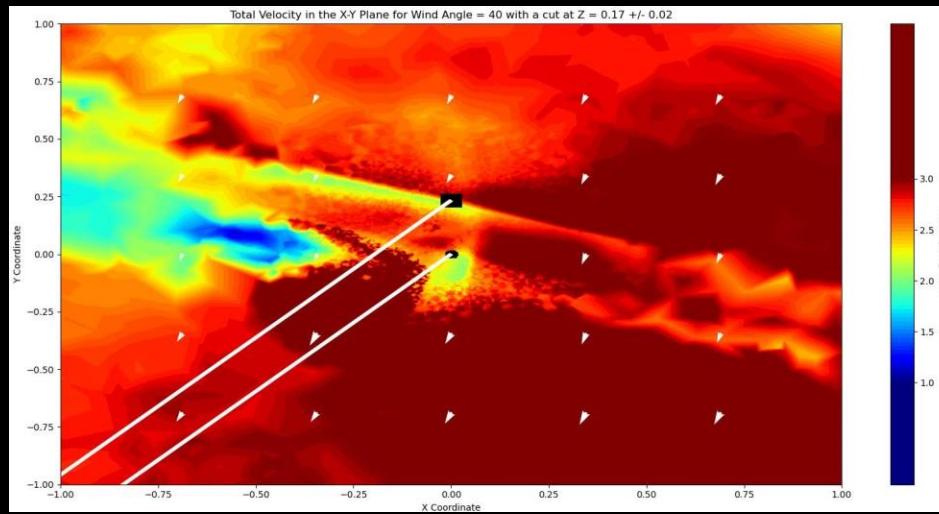


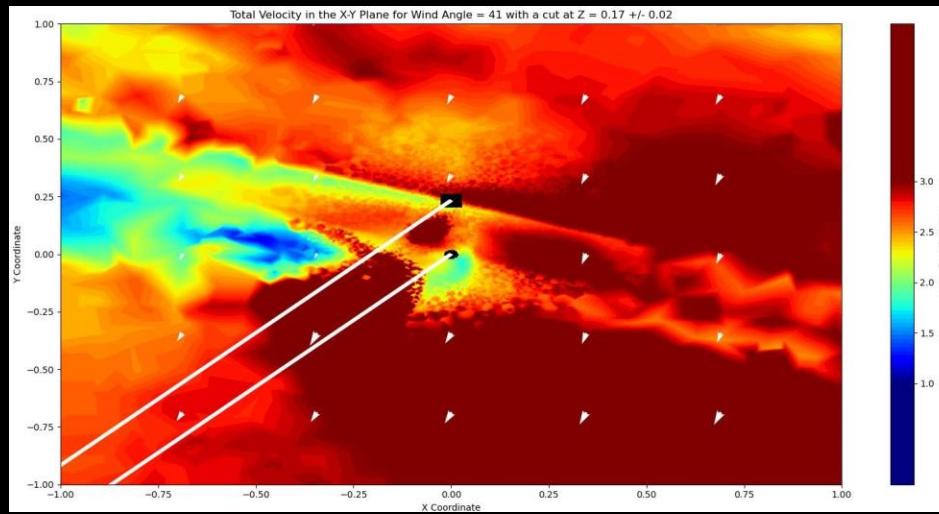


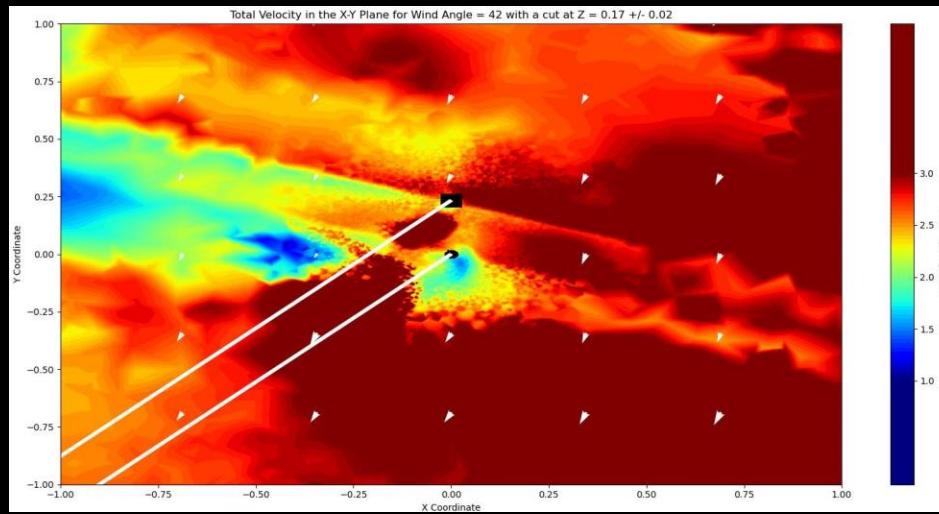


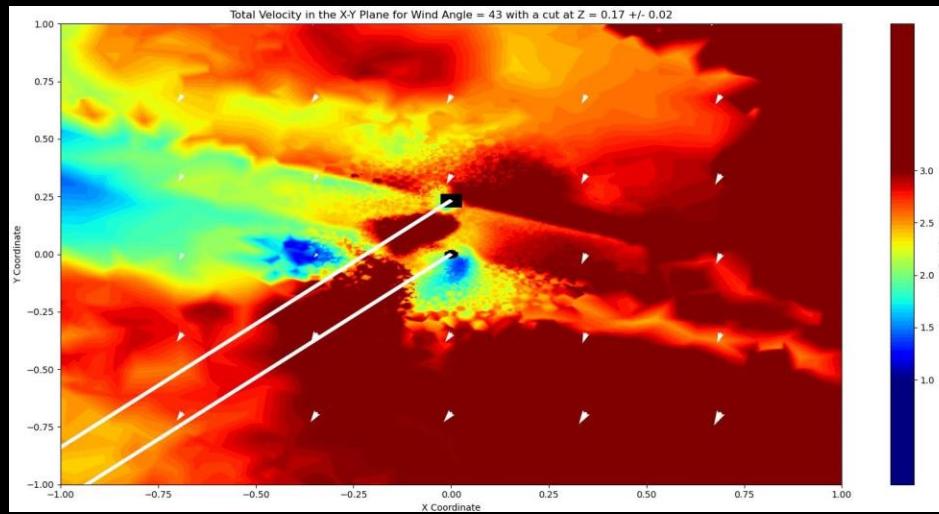


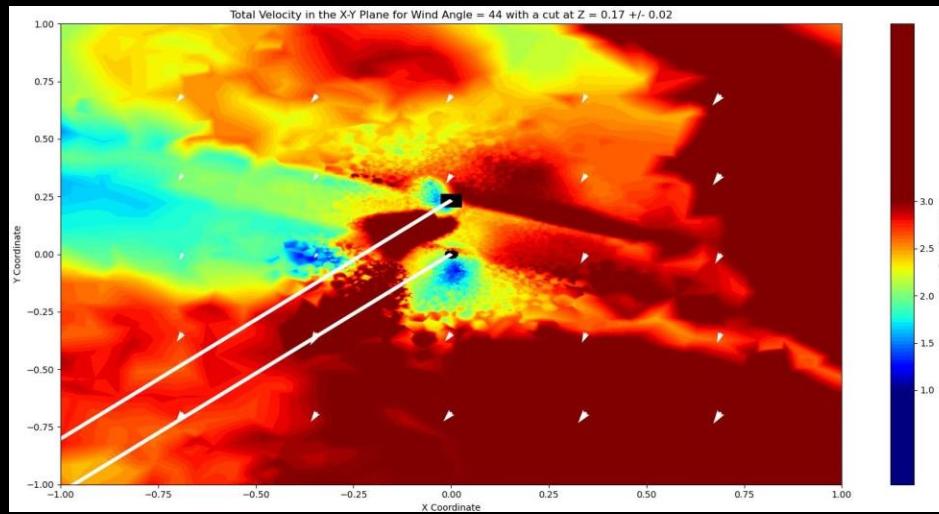


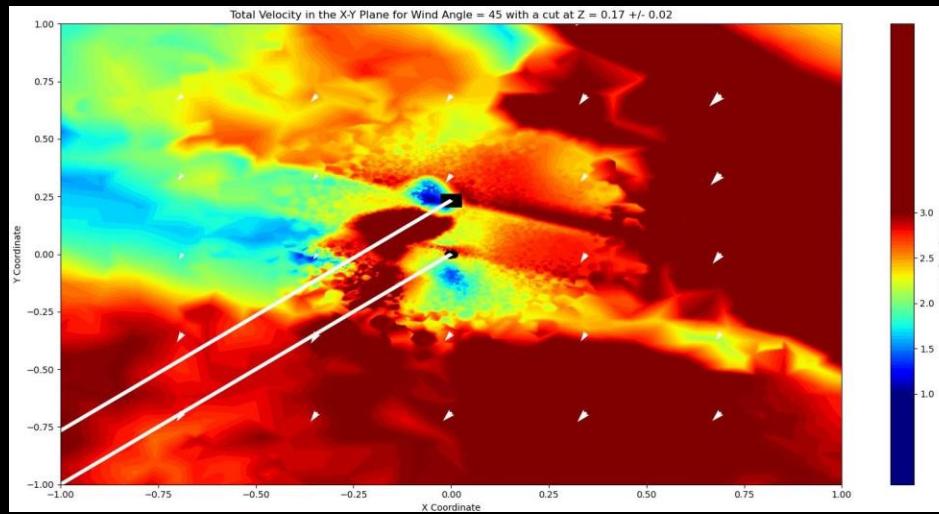


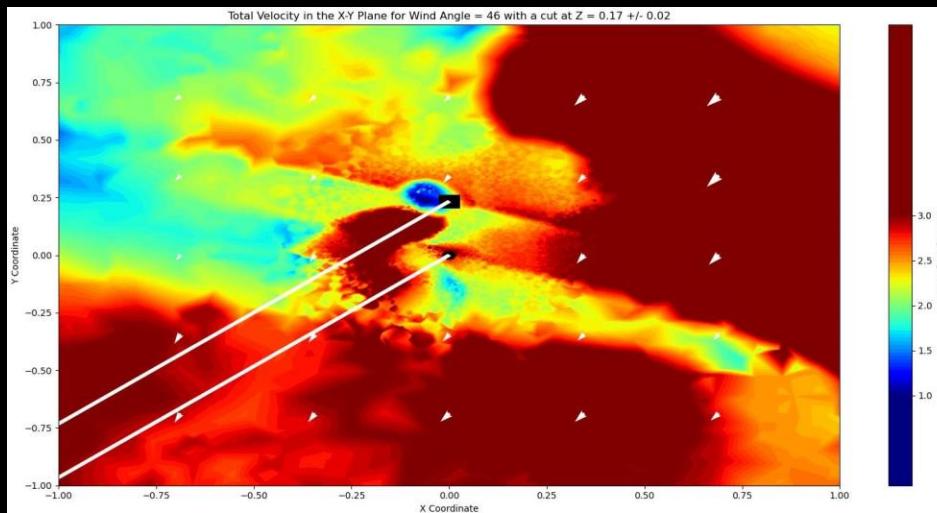


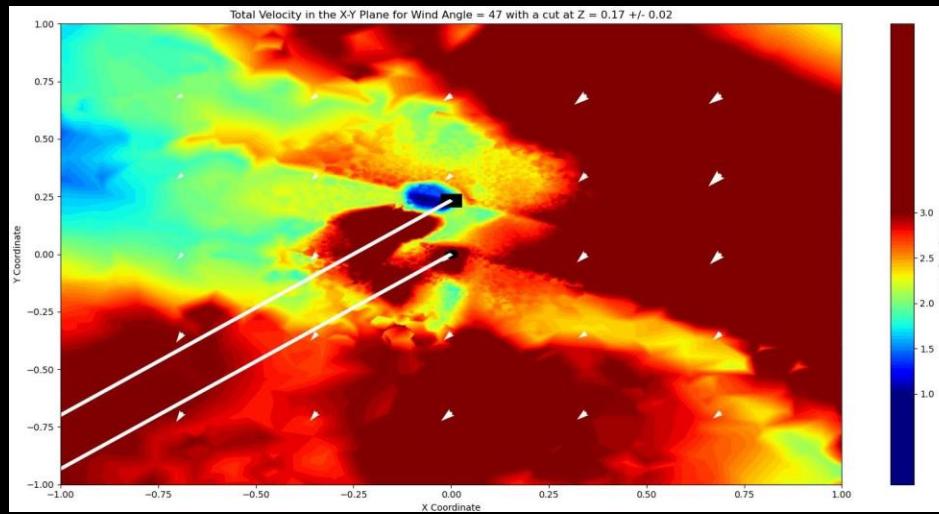


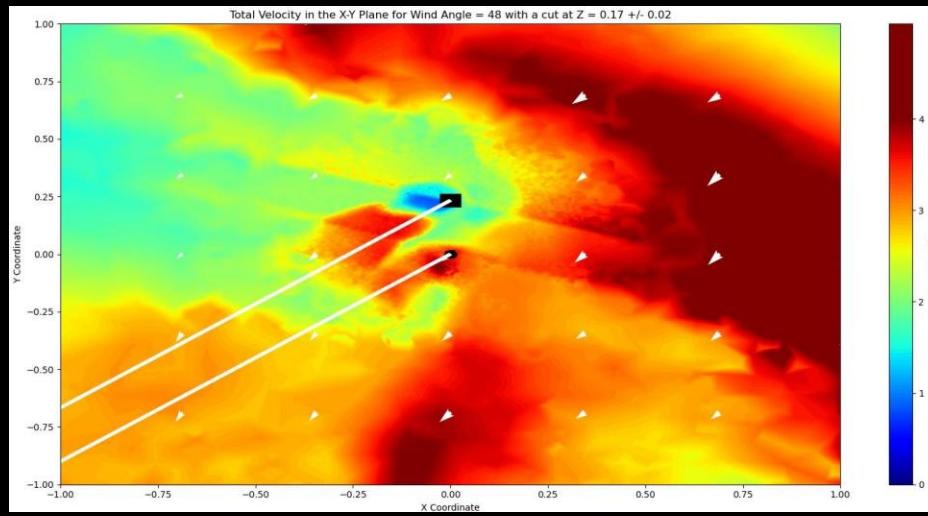


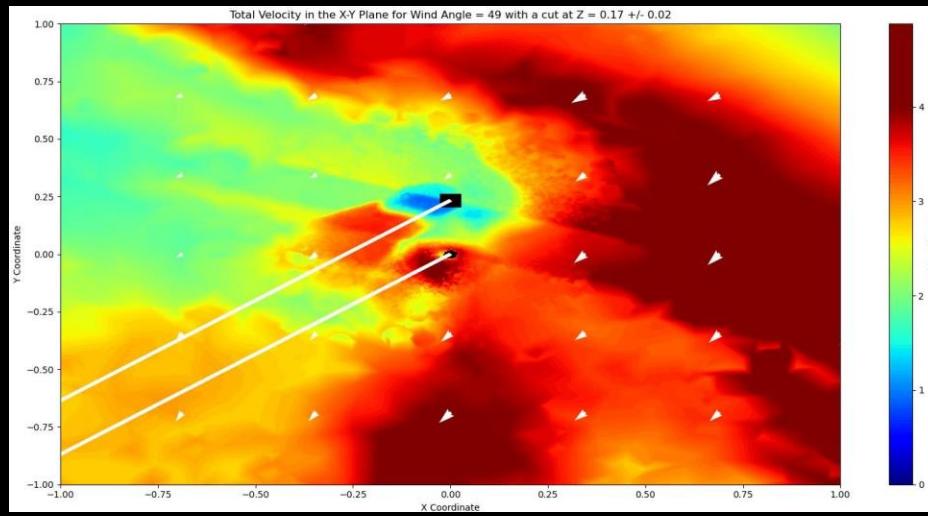


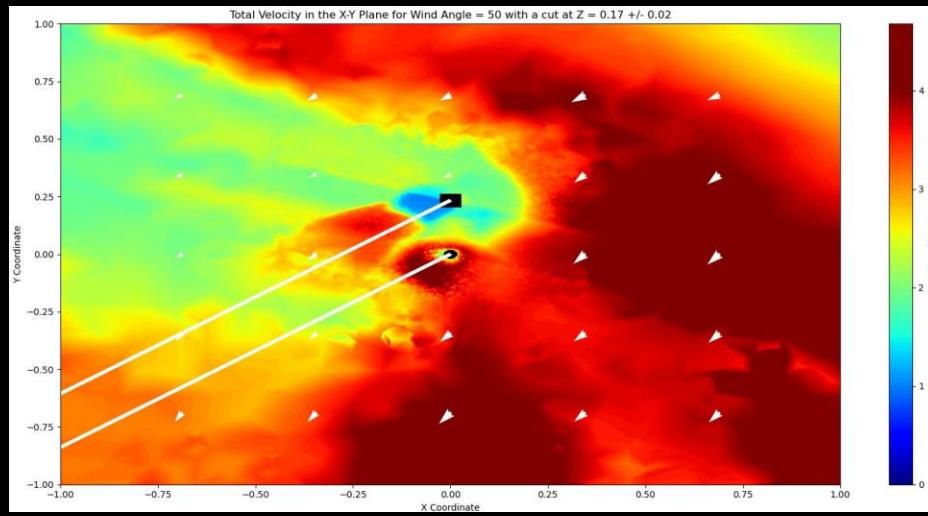


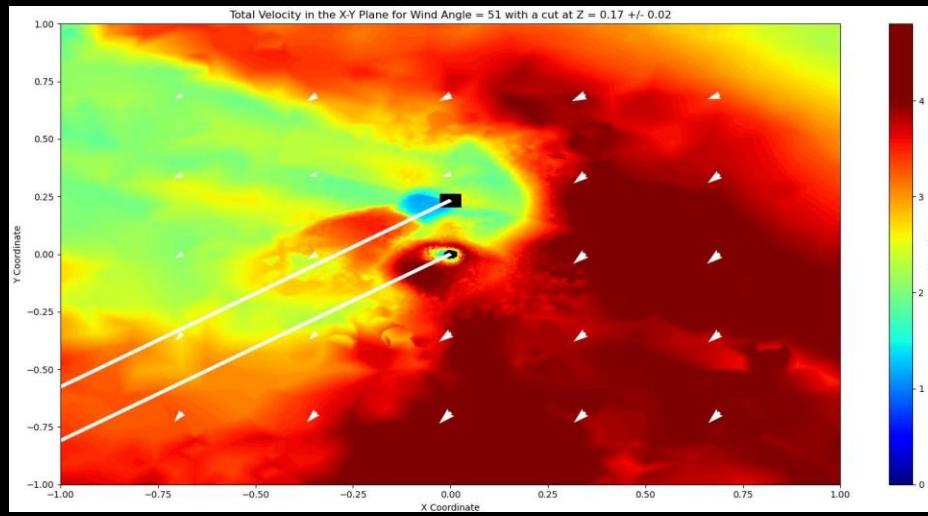


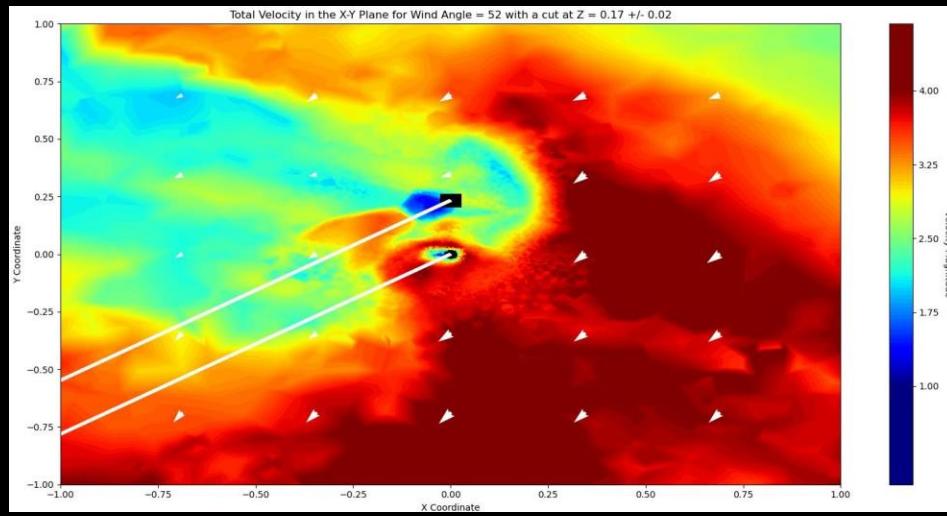


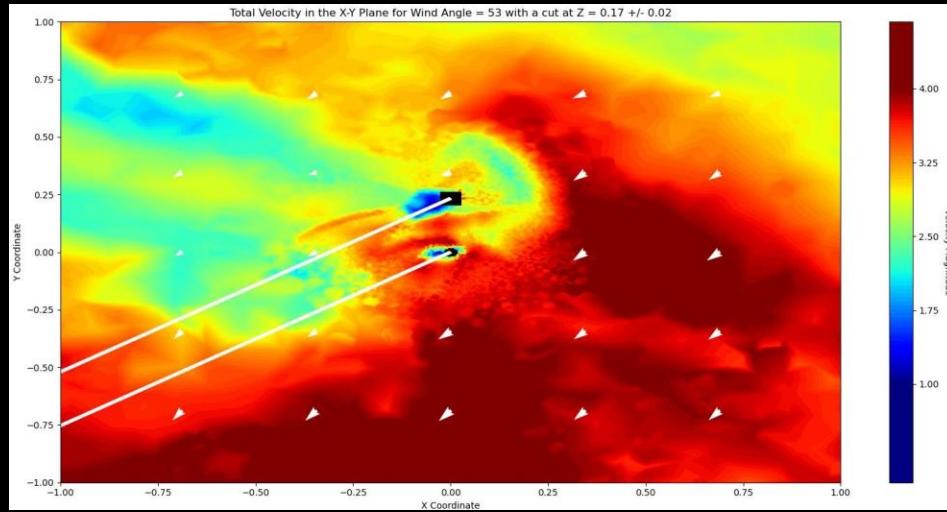


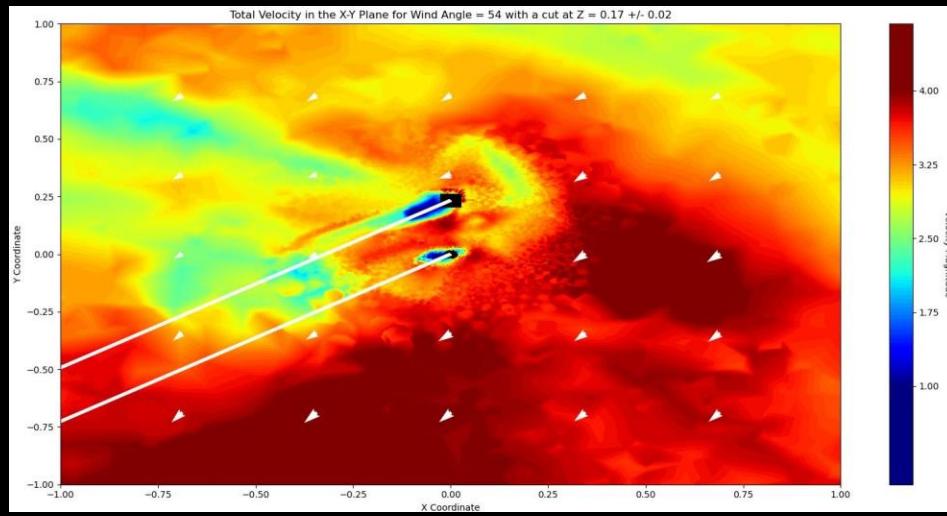


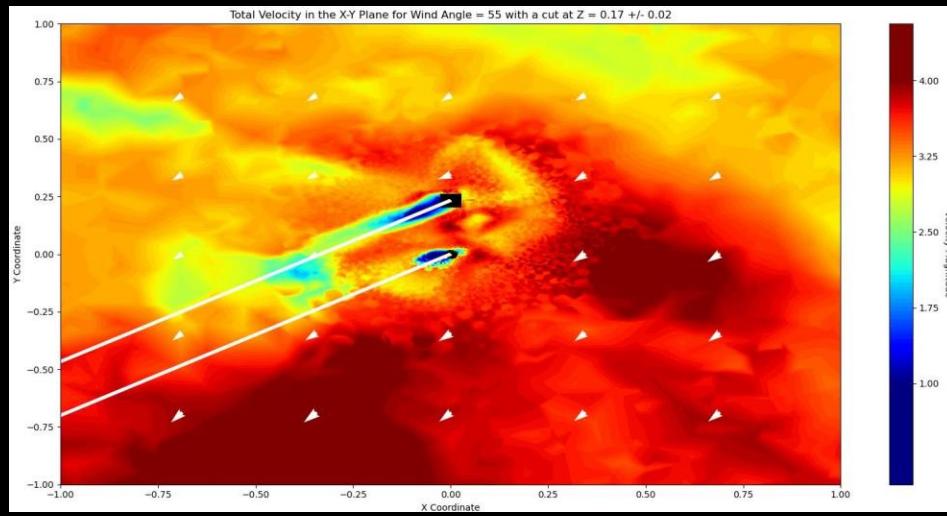


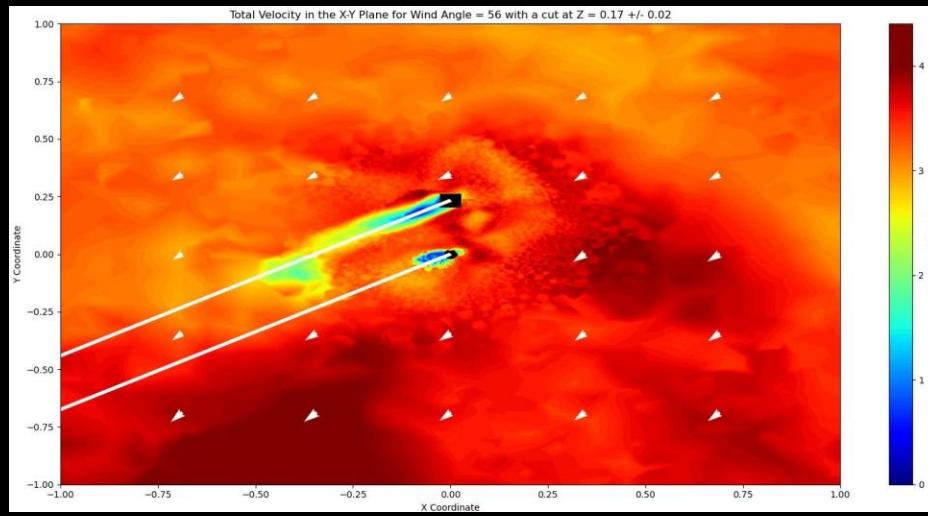


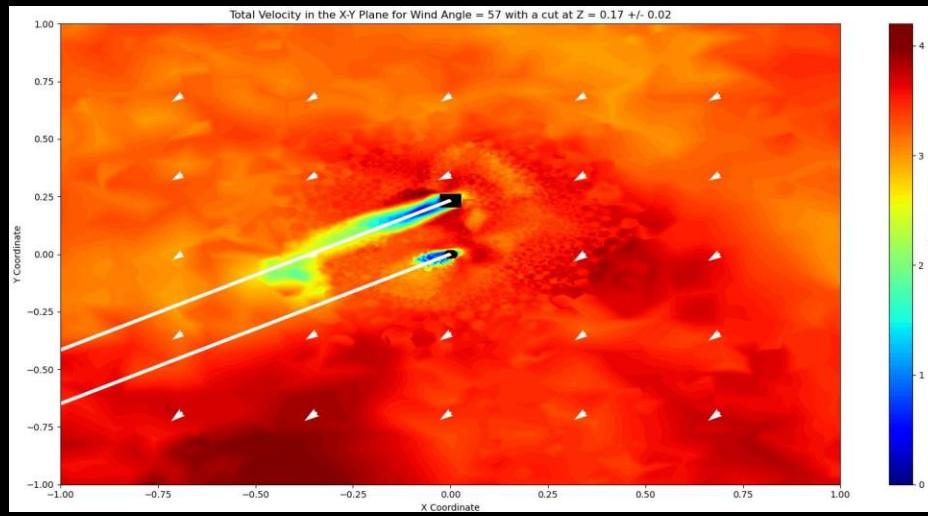


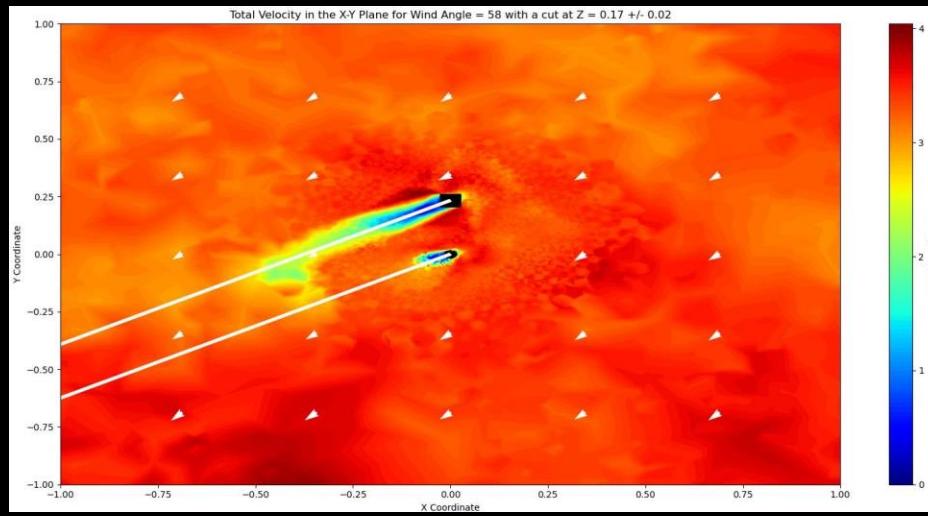


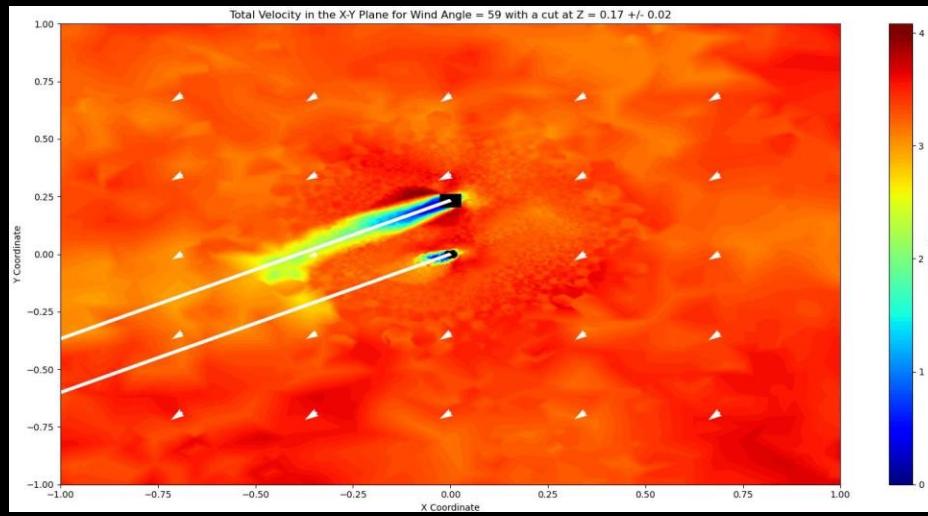


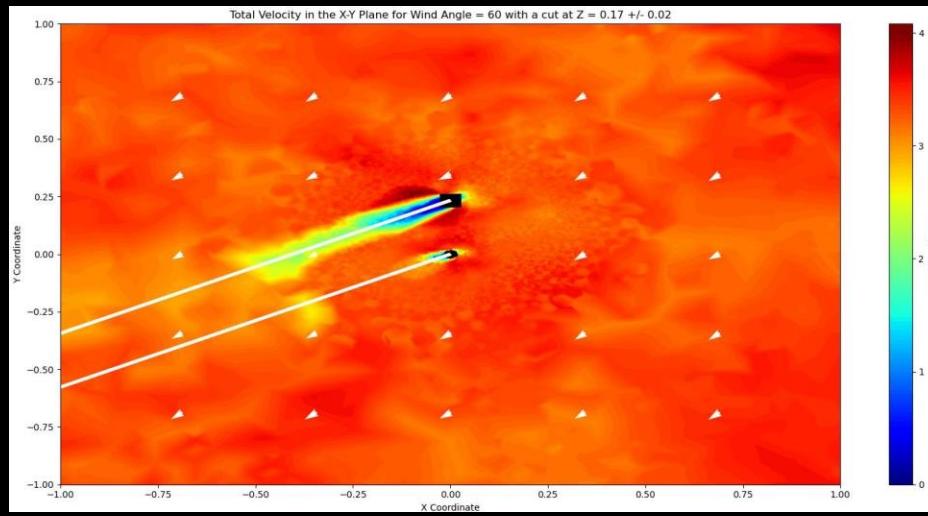


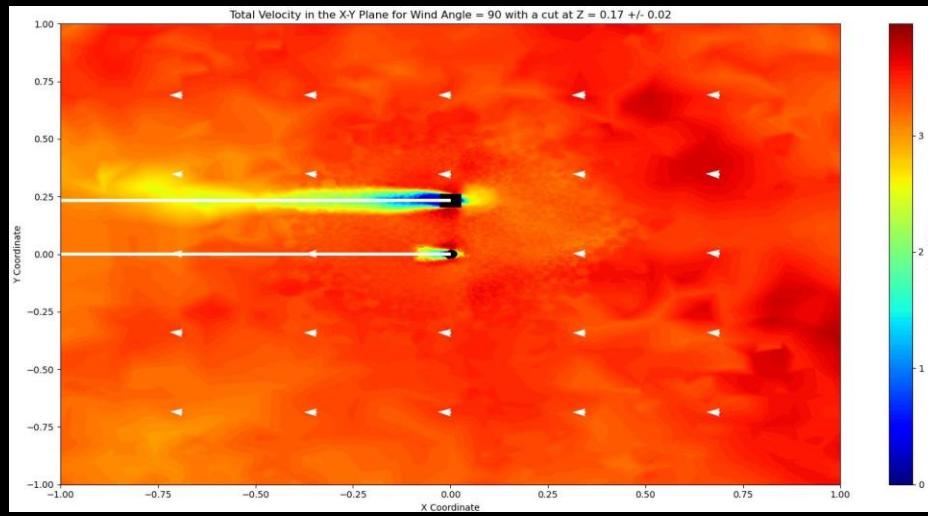


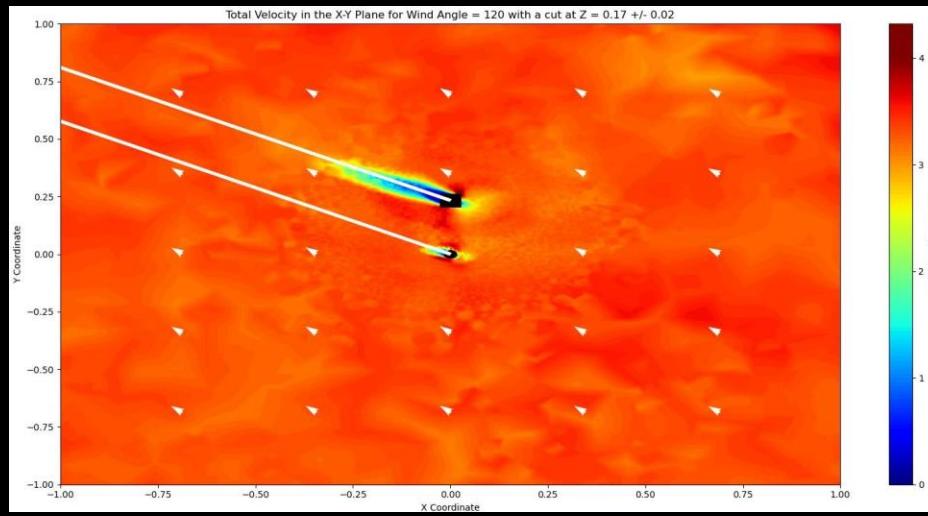


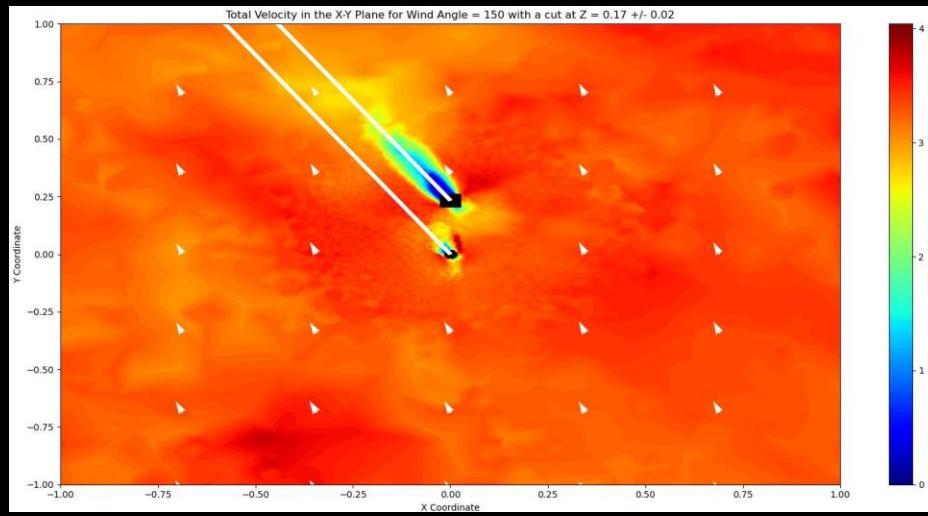


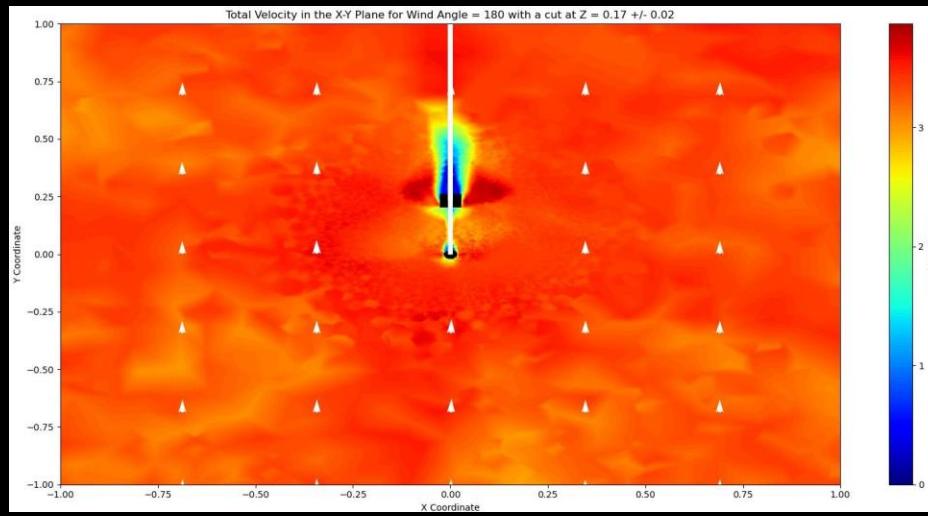


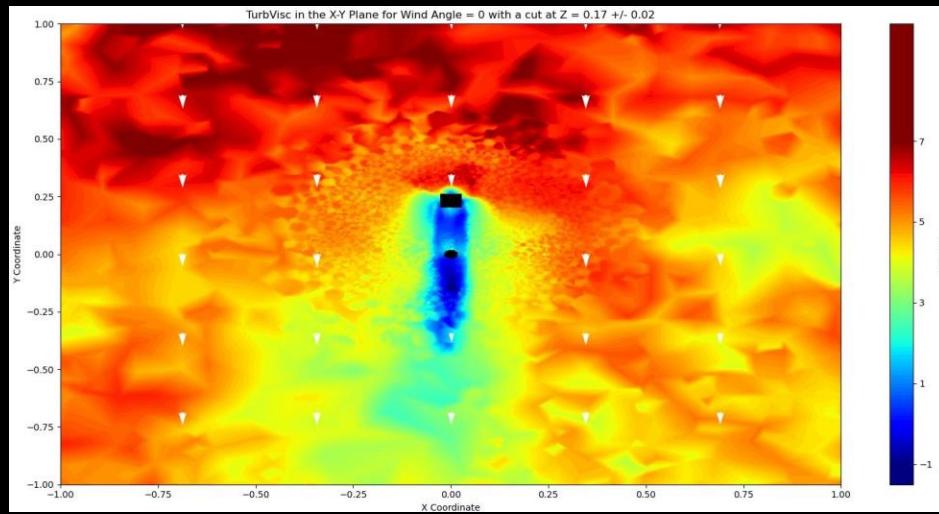


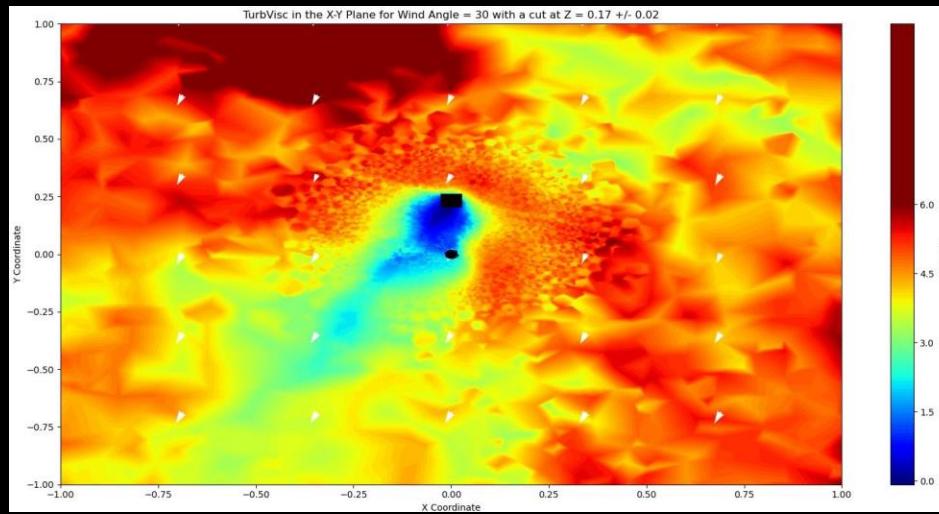


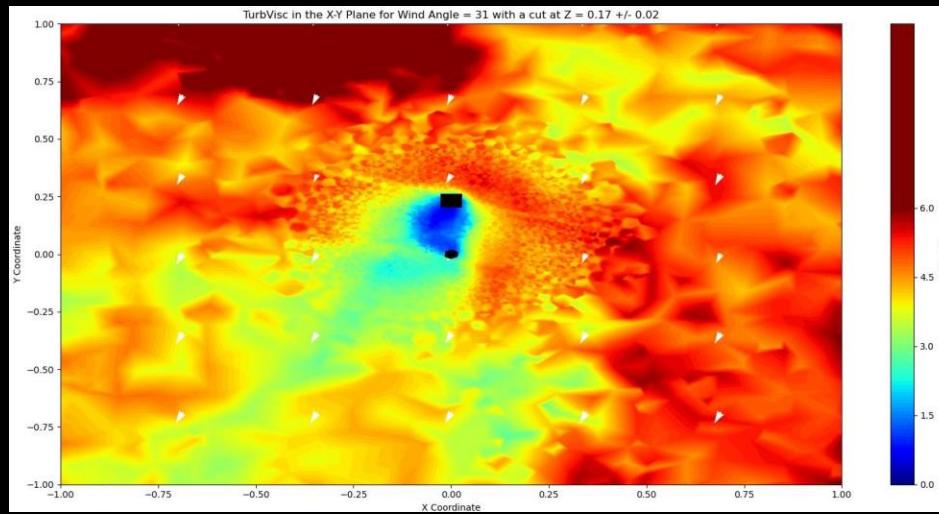


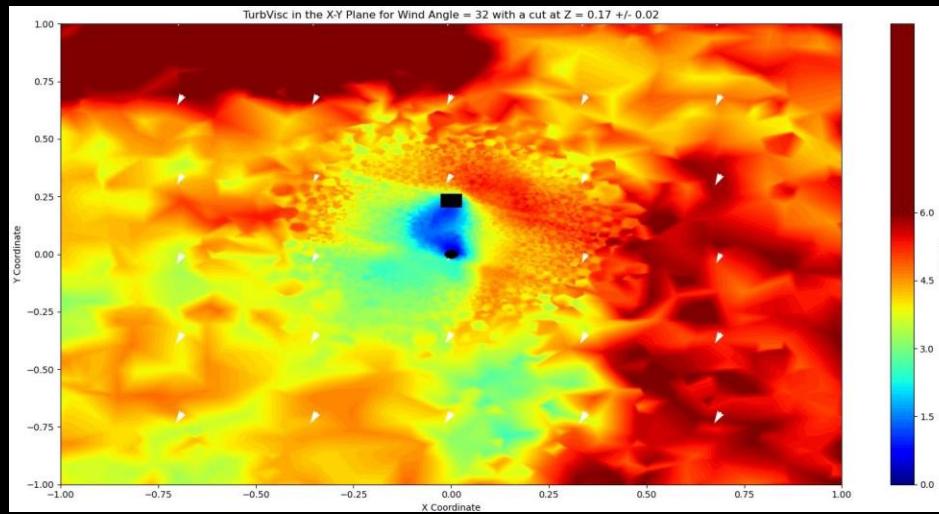


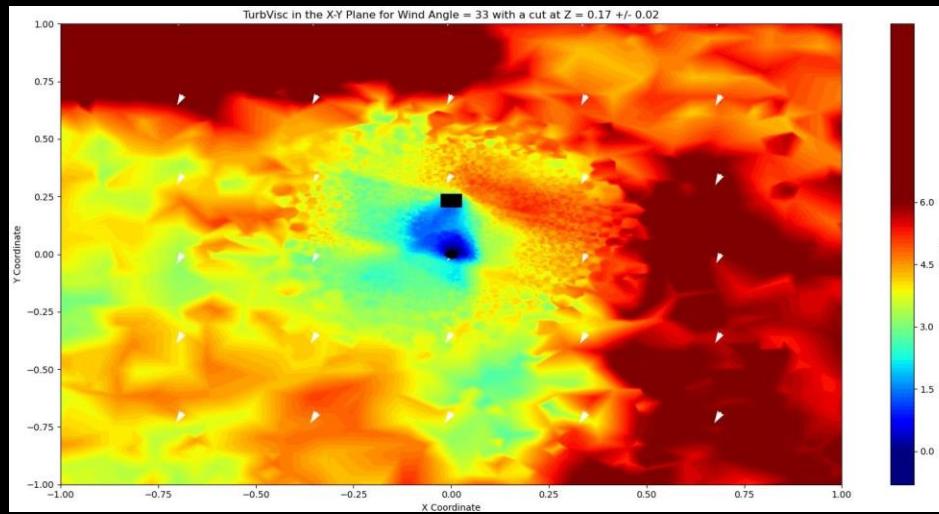


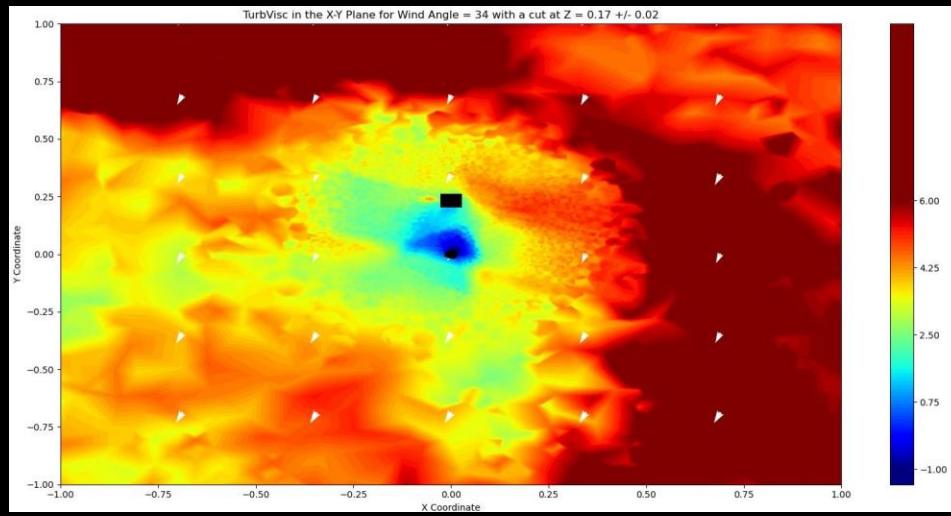


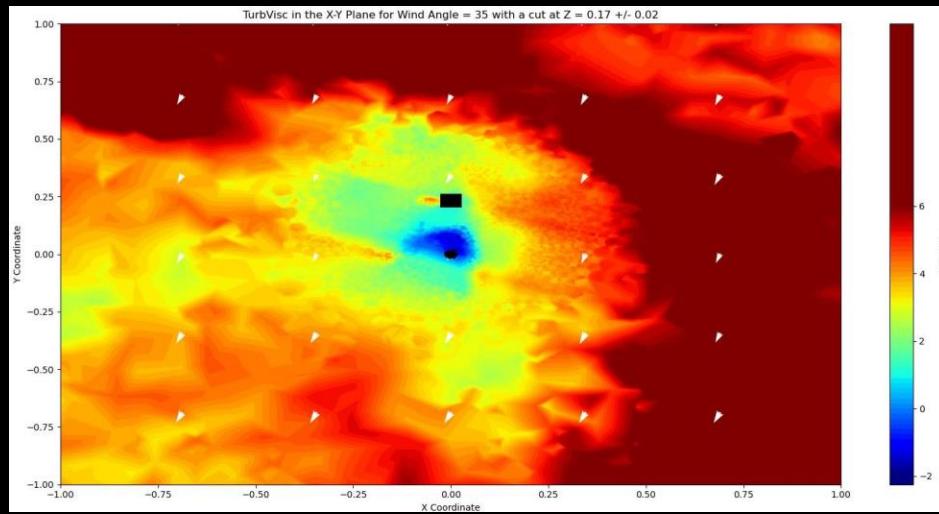


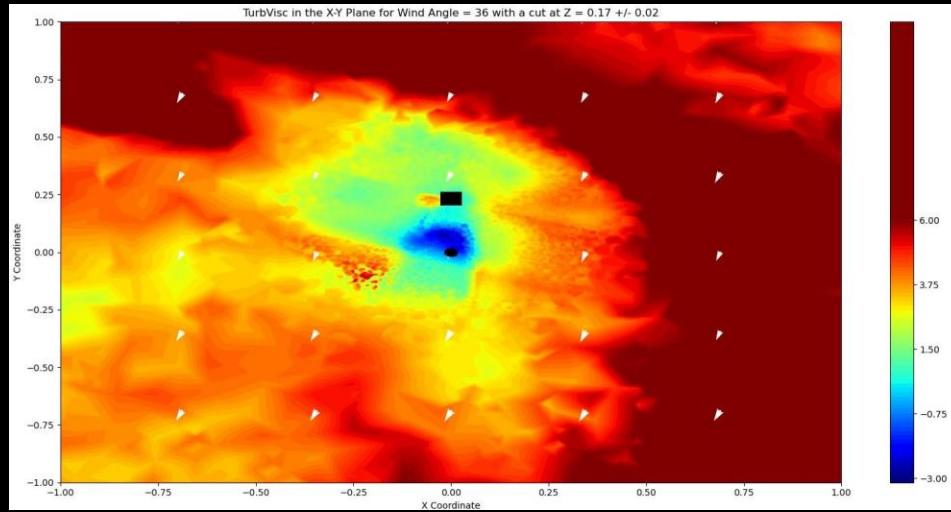


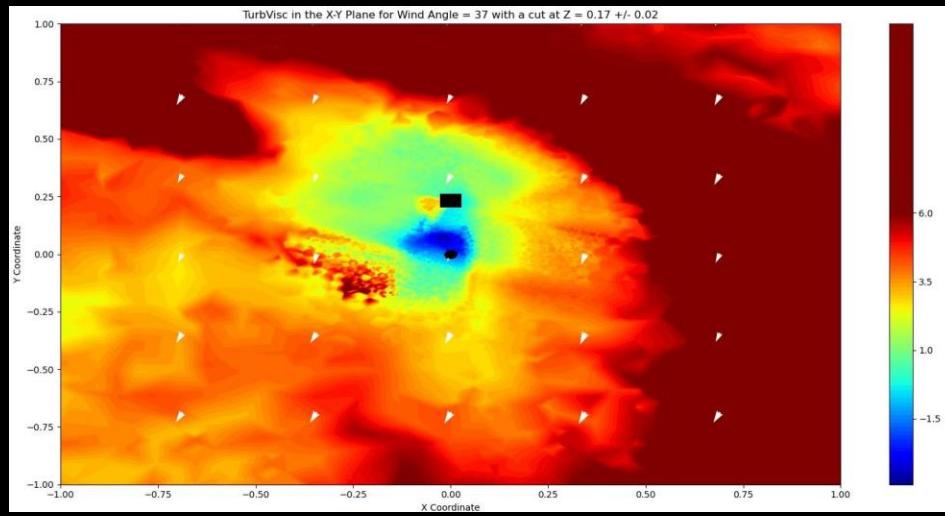


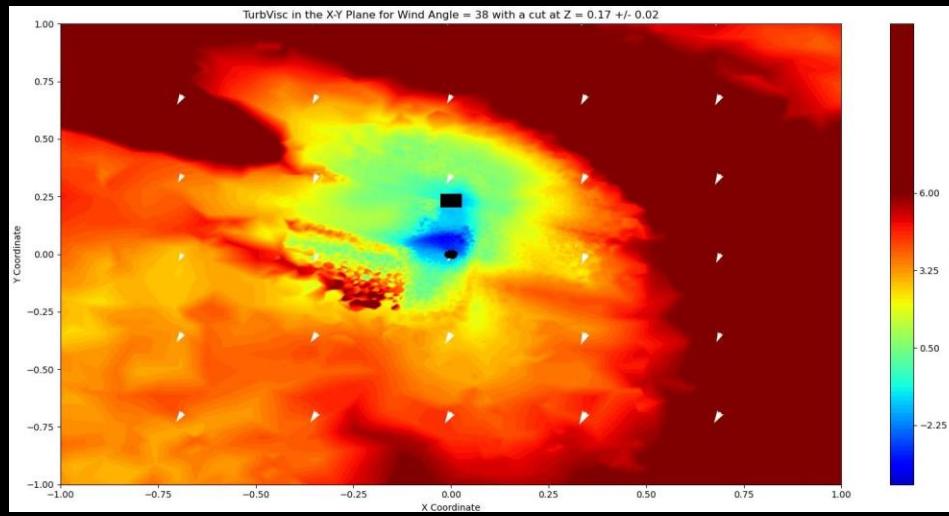


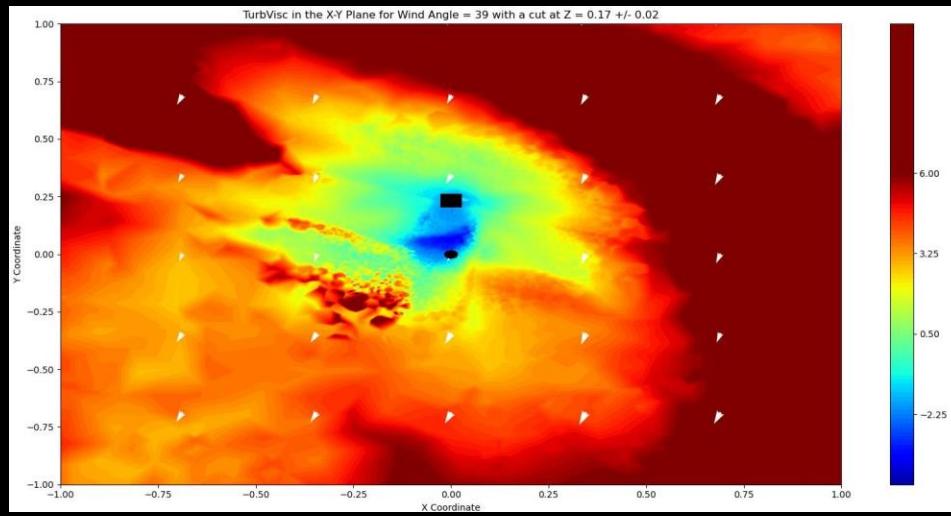


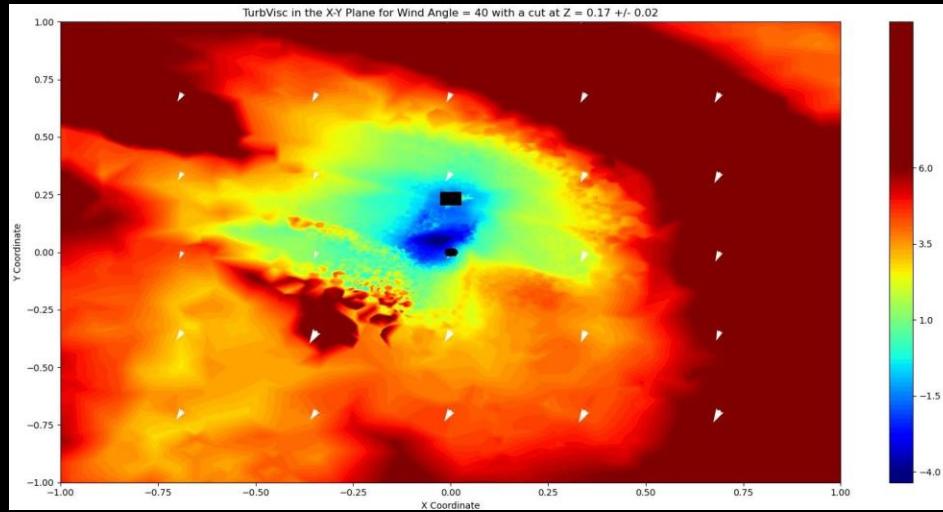


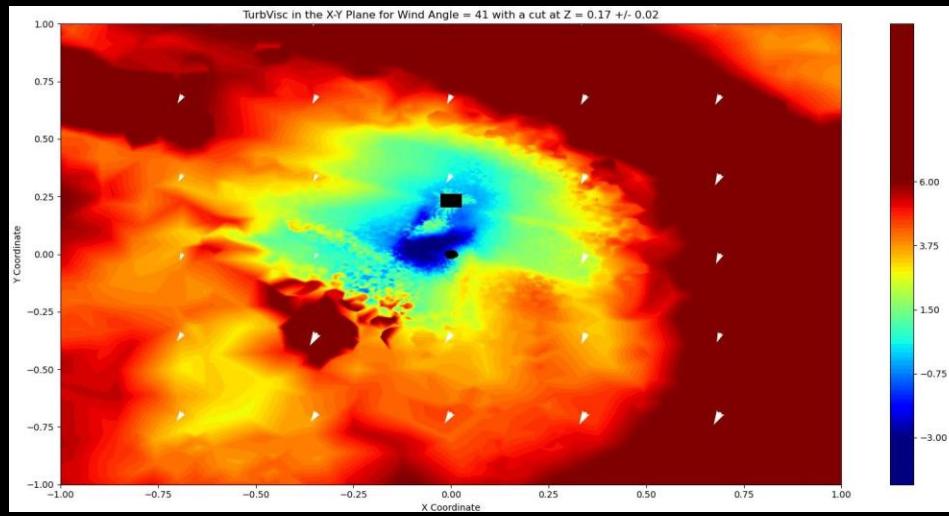


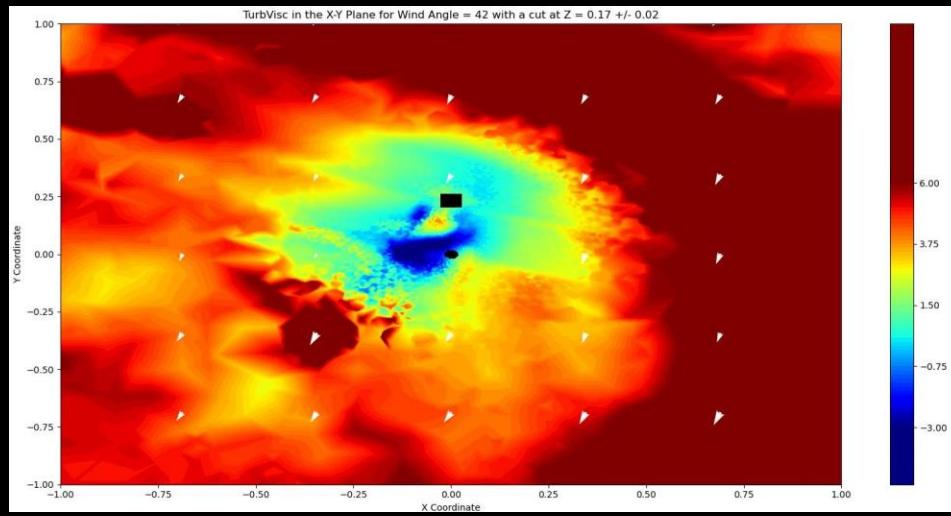


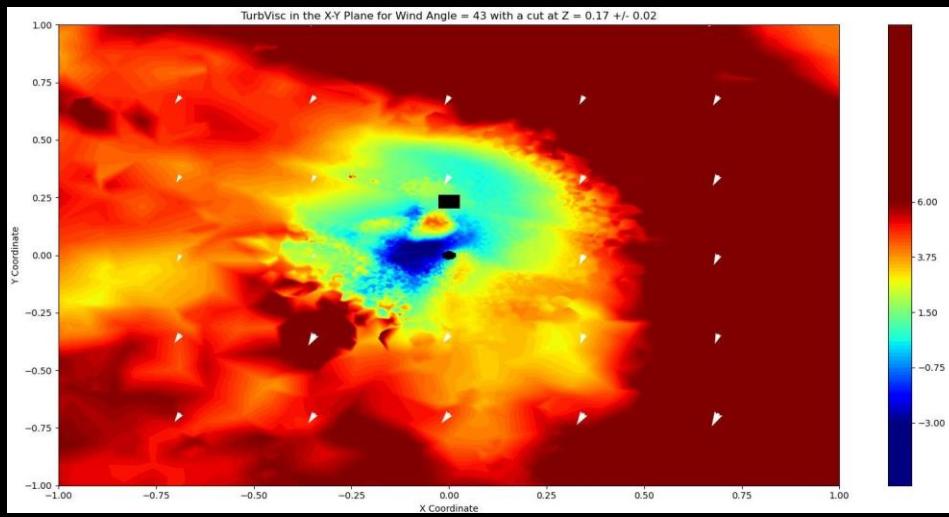


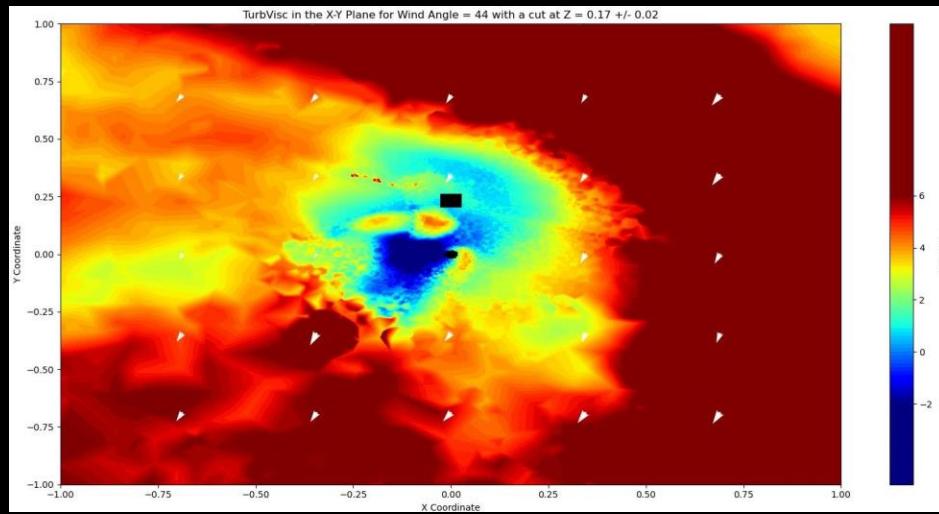


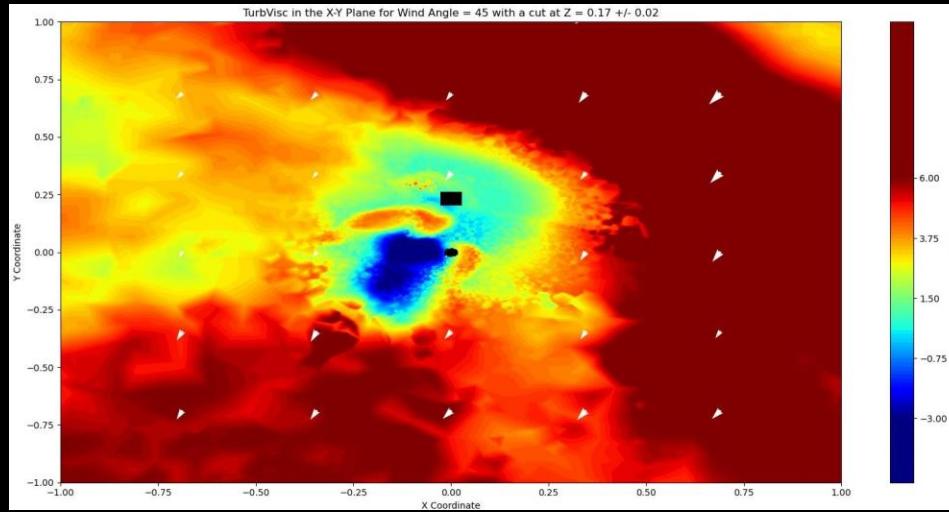


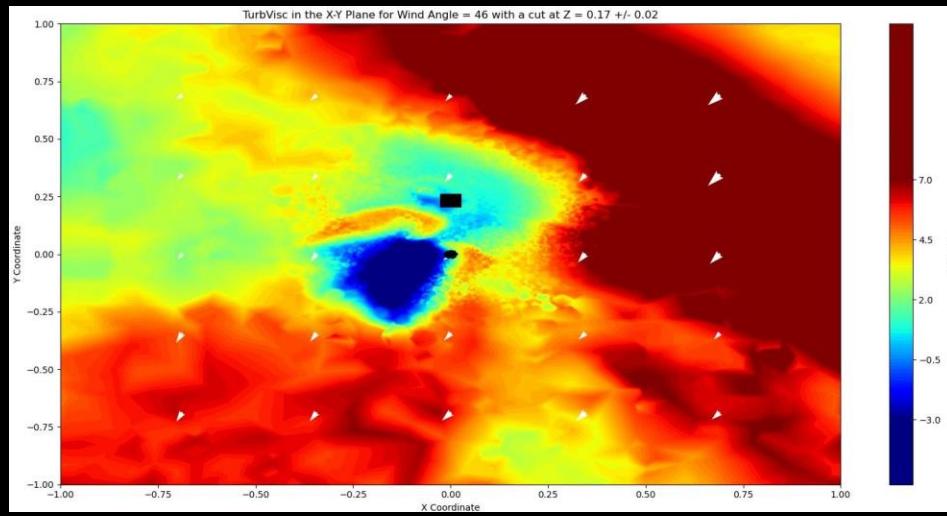


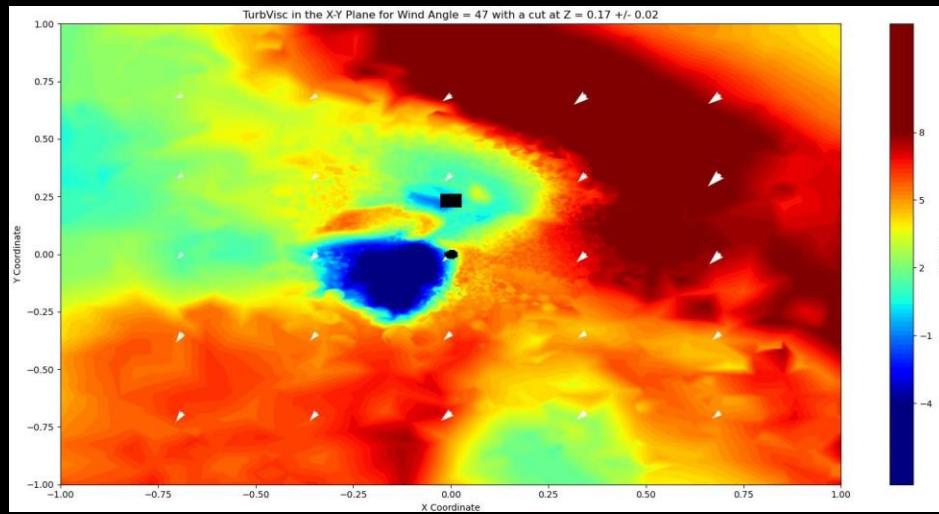


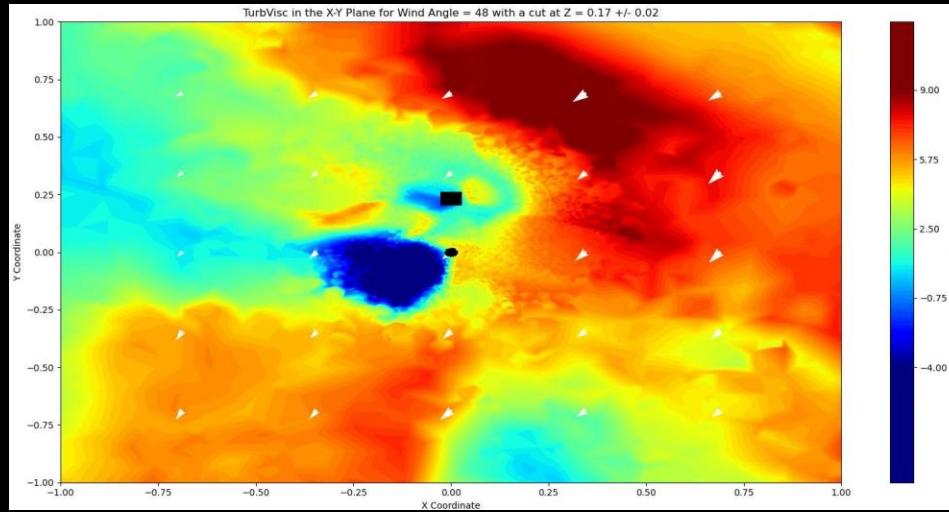


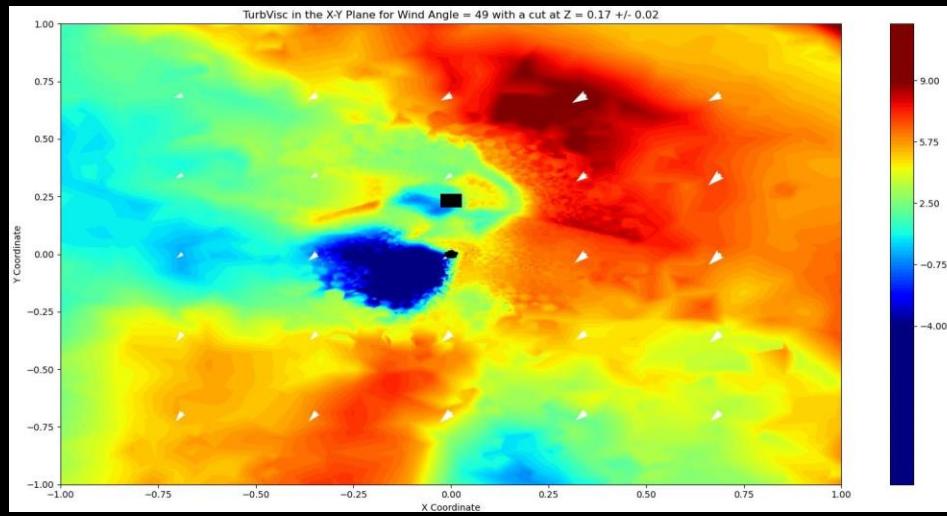


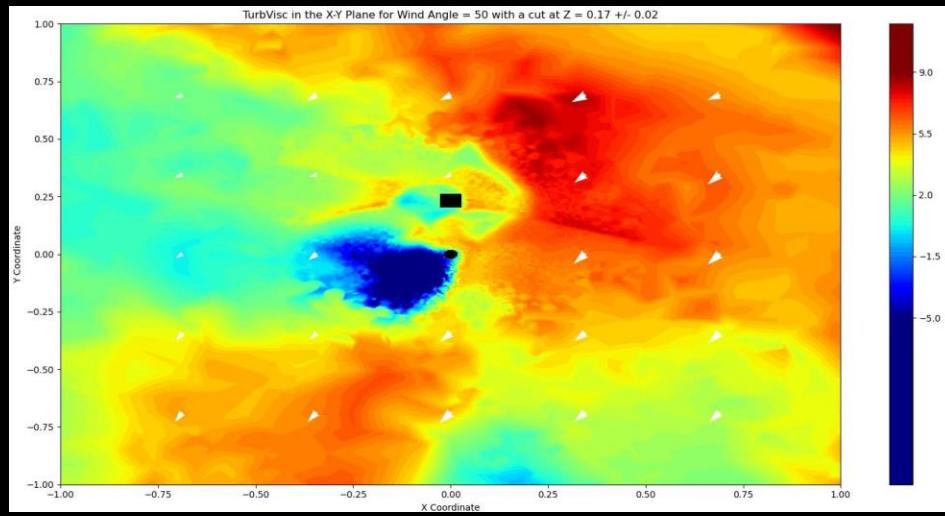


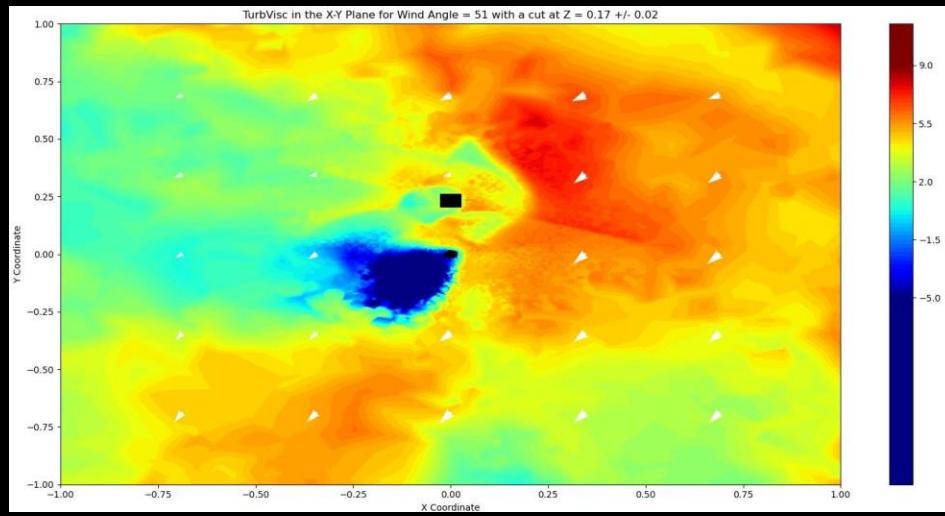


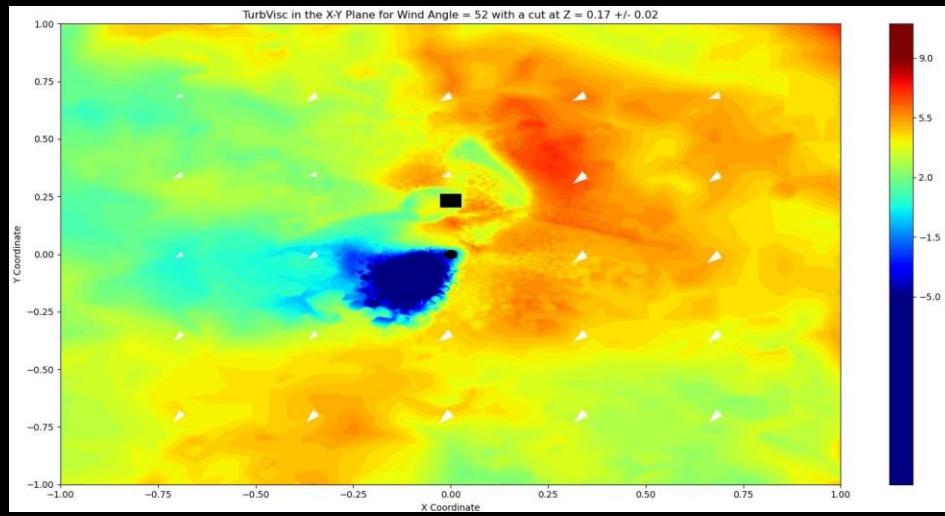


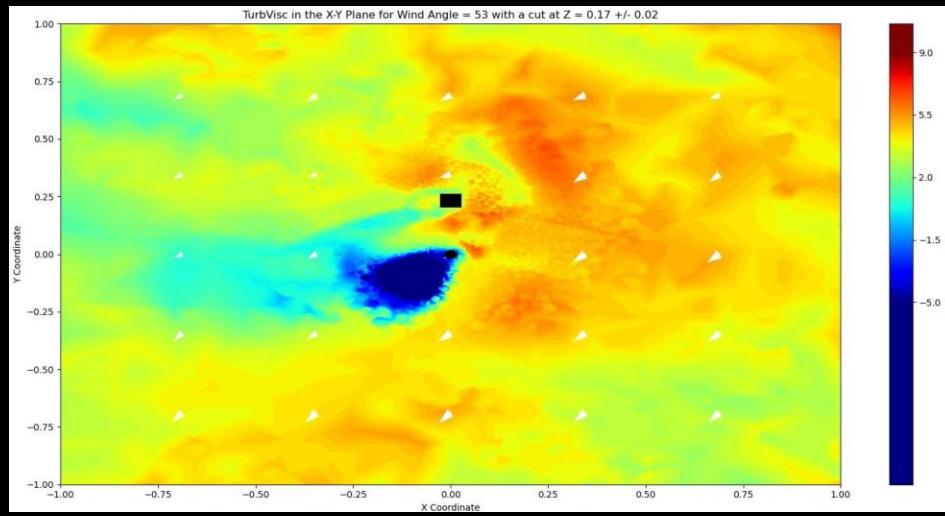


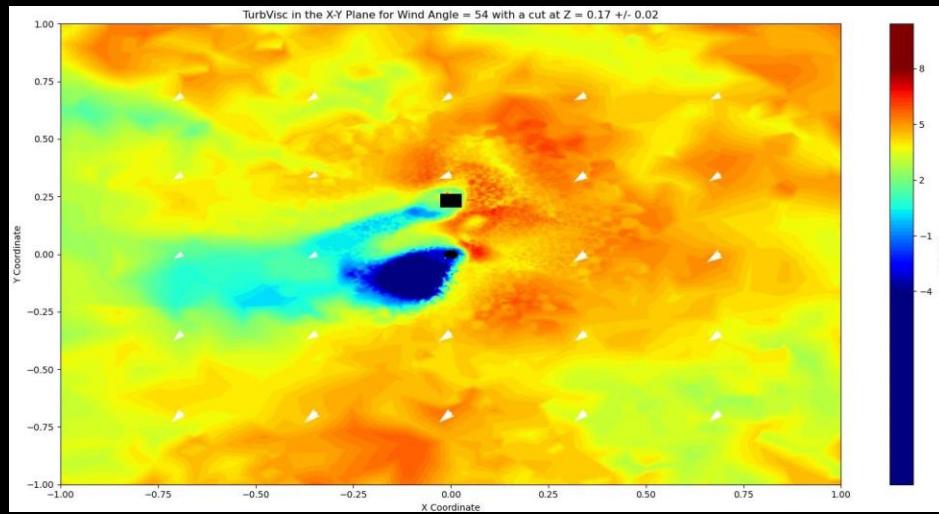


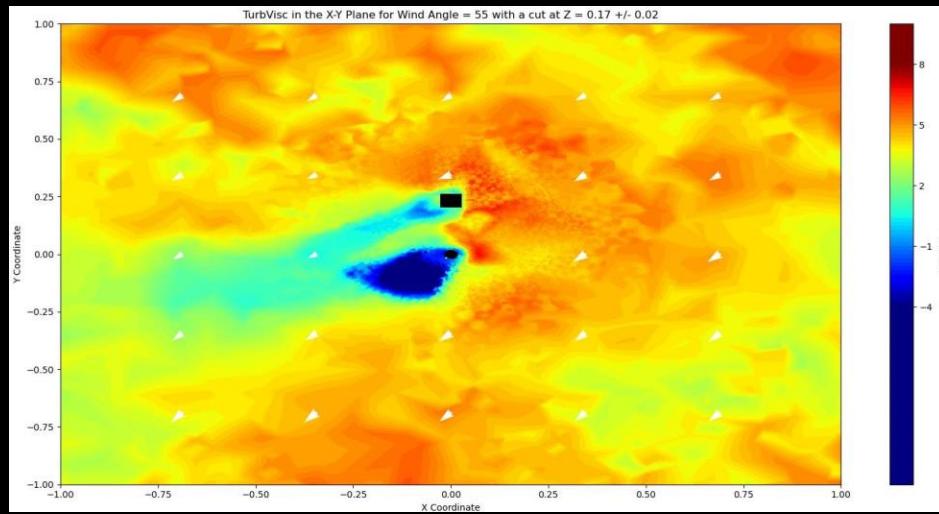


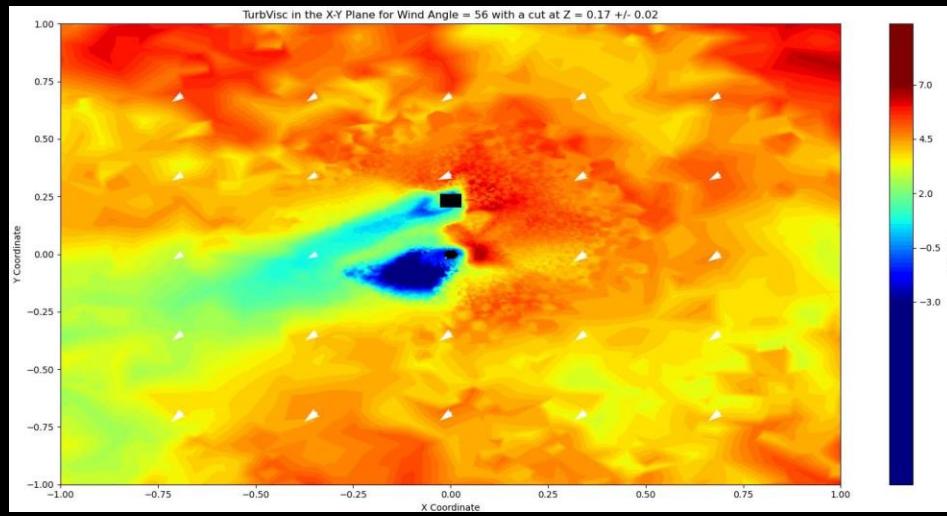


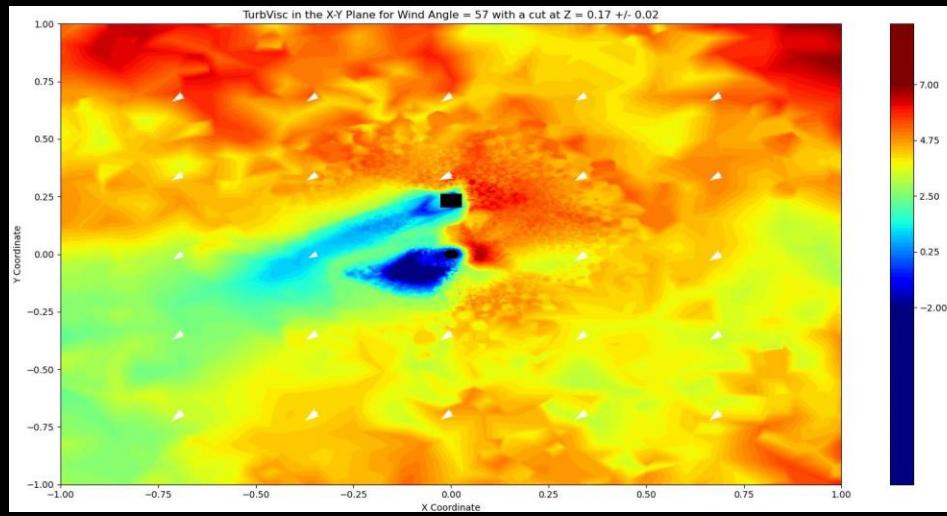


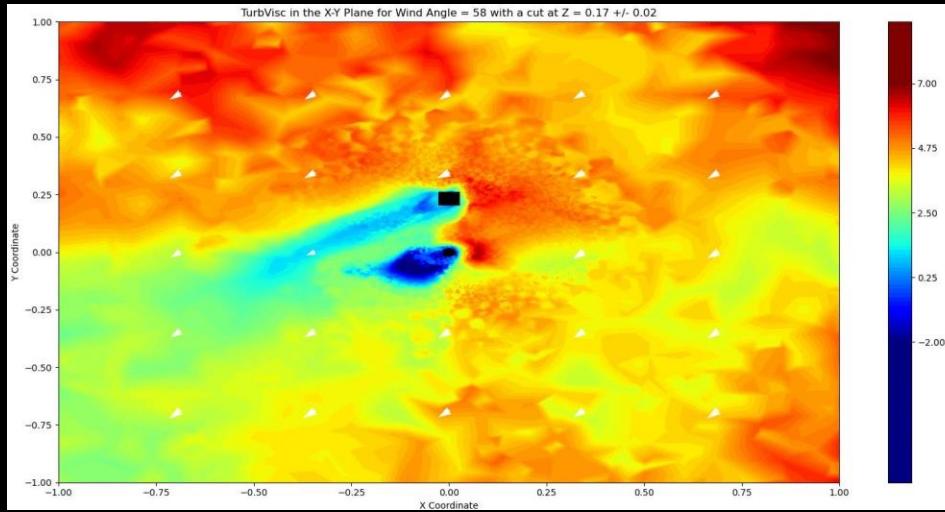


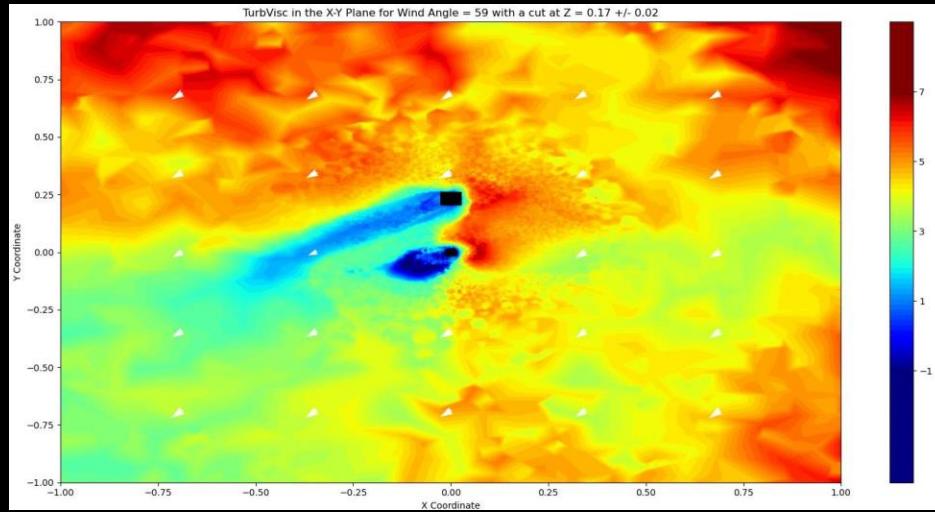


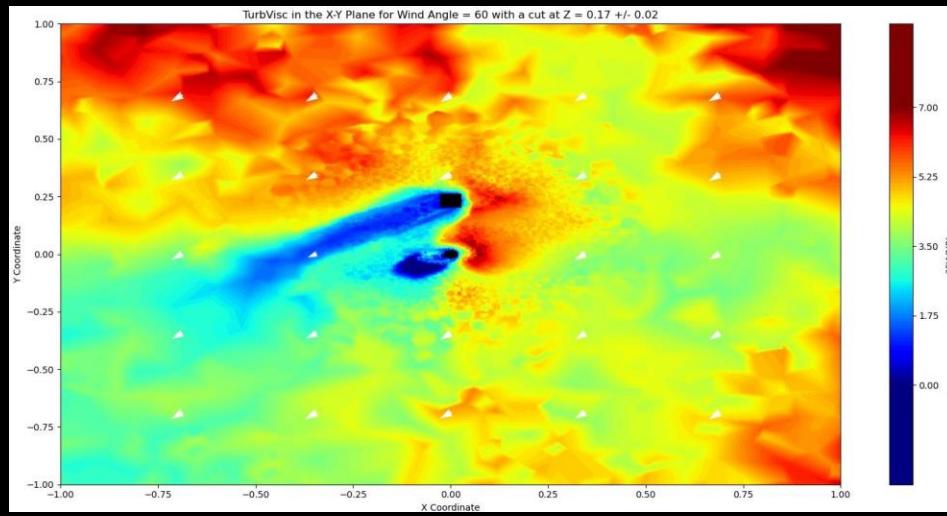


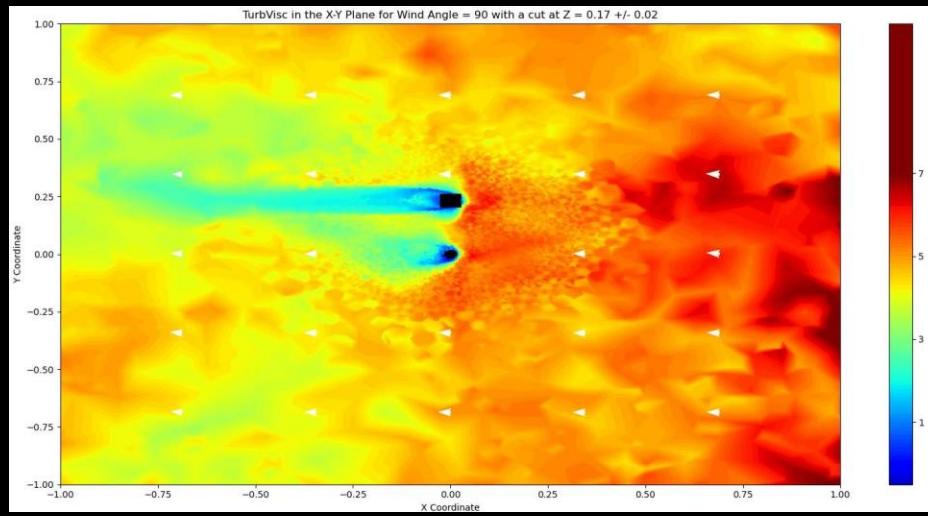


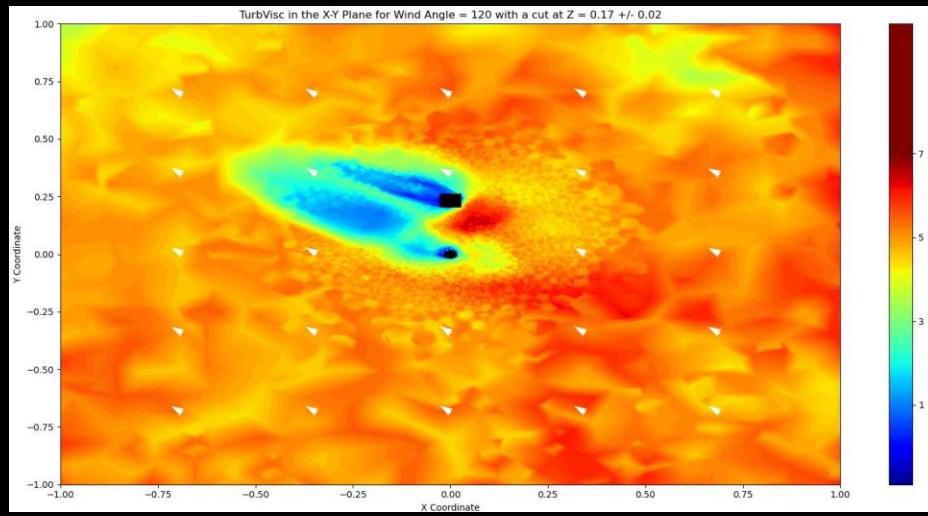


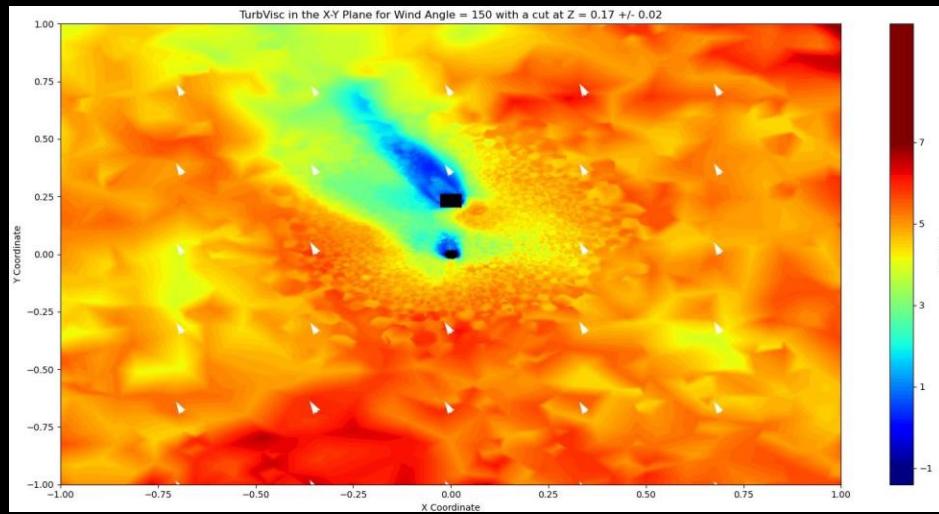


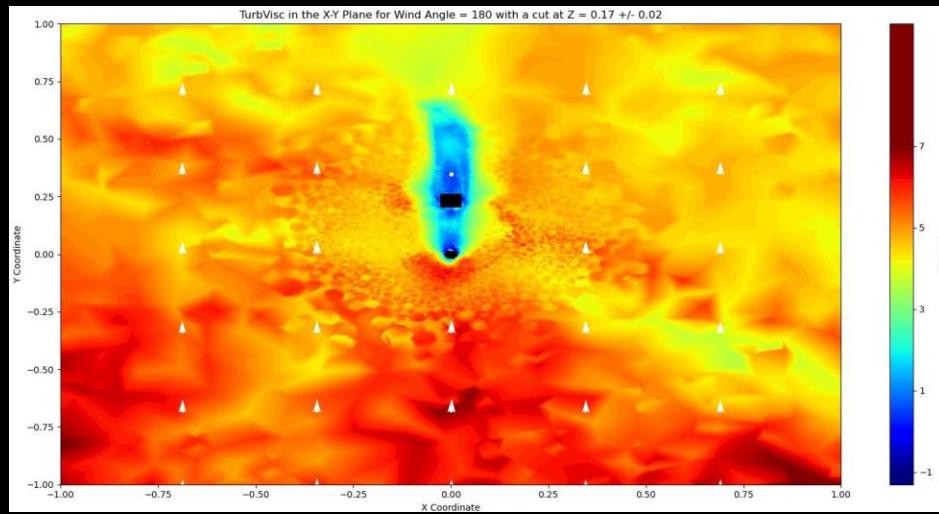












# Statistical Comparisons

## Statistical Comparisons - MSE

Type	Data – Std Normal (NC - 41790)	Data + Cont (NC - 8810)	Data + Cont + Boundary (NC - 2800)
Pressure	0.989688484	0.685449654	1.566548698
Velocity:0	0.321211131	0.118347663	0.152398519
Velocity:1	0.262792097	0.144505396	0.237096073
Velocity:2	0.002707722	0.007320889	0.01174556
TurbVisc	0.235428891	0.327236887	2.331652589

## Statistical Comparisons - MSE

Type	Data – Std Normal (NC - 27450)	Data + Cont (NC - 4660)	Data + Cont + Boundary (NC - 2595)
Pressure	0.439028	0.264865	2.34936
Velocity:0	0.7123	0.050552	0.144433
Velocity:1	0.758113	0.16956	0.249833
Velocity:2	0.9216	0.004823	0.011697
TurbVisc	0.998532	0.195052	2.276149

## Statistical Comparisons – R2

Type	Data – Std Normal (NC - 41790)	Data + Cont (NC - 8810)	Data + Cont + Boundary (NC - 2800)
Pressure	0.414511051	0.594495436	0.07324682
Velocity:0	0.684848359	0.883884907	0.850476404
Velocity:1	0.744623474	0.859572314	0.769594397
Velocity:2	0.917230623	0.776215818	0.640962943
TurbVisc	0.998284438	0.997615436	0.983009331

## Statistical Comparisons – R2

Type	Data – Std Normal (NC - 27450)	Data + Cont (NC - 4660)	Data + Cont + Boundary (NC - 2595)
Pressure	0.480088	0.843309	-0.38986
Velocity:0	0.726983	0.950401	0.858292
Velocity:1	0.792097	0.835225	0.757216
Velocity:2	0.918326	0.852572	0.642446
TurbVisc	0.998113	0.998579	0.983414

# Some Next Steps

Including RANS Momentum Loss

Convert scripts for TPU use

Modify boundary loss (Penalty Method as discussed in PM005)

Code Saturne Data for new angles – [15, 45, 75, 105, 165] (for Xiasu)

RANS

$$\frac{\partial \langle U_j \rangle}{\partial t} + \langle U_i \rangle \frac{\partial \langle U_j \rangle}{\partial x_i} = \frac{\partial}{\partial x_i} \left[ \nu_{\text{eff}} \left( \frac{\partial \langle U_i \rangle}{\partial x_j} + \frac{\partial \langle U_j \rangle}{\partial x_i} \right) \right] - \frac{1}{\rho} \frac{\partial}{\partial x_j} \left( \langle p \rangle + \frac{2}{3} \rho k \right)$$

# TPU vs GPU

A TPU, or Tensor Processing Unit, is a type of application-specific integrated circuit (ASIC) developed by Google for neural network machine learning. It is specifically designed to accelerate machine learning tasks. Here are some key differences between a TPU and a GPU:

## 1. Design Purpose:

TPU: Optimized for machine learning and deep learning tasks.

GPU: Designed for graphics processing and rendering but has been adapted for parallel processing tasks, including machine learning.

## 2. Architecture:

TPU: Features a matrix processing unit for handling large amounts of matrix operations, which are common in deep learning.

GPU: Utilizes many small, efficient cores designed for handling multiple tasks simultaneously, suitable for both graphics and general parallel computing tasks.

## 3. Performance:

TPU: Highly efficient for specific tasks like neural network inference and training, offering high throughput for matrix calculations.

GPU: Offers versatile performance for a broader range of computing tasks, including but not limited to AI and machine learning.

# TPU vs GPU

## 4. Flexibility:

TPU: More specialized and less flexible.

GPU: More flexible in terms of the variety of computations it can perform, useful in gaming, graphics, and general purpose computing.

## 5. Integration and Accessibility:

TPU: More commonly found in cloud-based environments (like Google Cloud/CoLab) and less accessible for consumer-level usage.

GPU: Widely available for both consumer and enterprise use, with a broad range of applications beyond machine learning.

## 6. Software Compatibility:

TPU: Optimized for specific frameworks and may require specialized software to fully utilize its capabilities.

GPU: Compatible with a wide range of software and programming models, including CUDA for NVIDIA GPUs, making it more adaptable to various programming needs.

## 7. Cost and Availability:

TPU: Generally more expensive and less available to the average consumer, often accessed through cloud services.

GPU: More widely available and comes in a range of price points, from consumer-grade to high-end enterprise models.